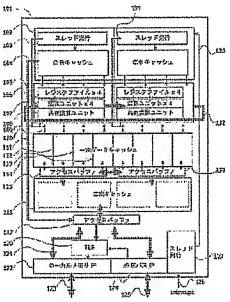
PIPE LINE PARALLEL PROCESSOR USING MULTI-THREAD

Publication number: JP2001236221 (A)
Publication date: 2001-08-31
Inventor(s): SHINDO KEISUKE +
Applicant(s): SHINDO KEISUKE +
Classification:
- international: ***G06F12/08; G06F12/10; G06F12/12; G06F9/30; G06F9/34; G06F9/38; G06F9/46;*** G06F12/08; G06F12/10; G06F12/12; G06F9/30; G06F9/34; G06F9/38; G06F9/46; (IPC1-7): G06F12/08, G06F12/10, G06F12/12, G06F9/30, G06F9/34, G06F9/38, G06F9/46
- European:
Application number: JP20000042696 20000221
Priority number(s): JP20000042696 20000221

Abstract of **JP 2001236221 (A)**

PROBLEM TO BE SOLVED: To establish both frequency performance and parallel performance by shortening memory witing in a system for successively operating plural threads by arithmetic units arranged in a row in a processor using a multi-thread program, and to prevent inter-node data transfer interrupting the parallel processing performance and waiting through synchronization. SOLUTION: Plural caches for storing data are loaded on a processor carried by patent gazette 1999-287662, and each cache is connected to several arithmetic executing units. The contents of the cache are transferred and duplicated according to the progress of threads. When the contents of the cache can not completely transferred, one thread is executed by a single arithmetic executing unit. Moreover, access to the designated address is detected by using a virtual storage mechanism and the shared mechanism of the caches, and the threads are resumed.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2001－236221
（P2001－236221A）
(43)公開日　平成13年８月31日(2001.8.31)

(51)Int.Cl.[7]		識別記号	ＦＩ			ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ	9/38	３１０	Ｇ０６Ｆ	9/38	３１０Ｅ	５Ｂ００５
					３１０Ａ	５Ｂ０１３
		３５０			３５０Ｘ	５Ｂ０３３
		３７０			３７０Ｘ	５Ｂ０９８
	9/30	３５０		9/30	３５０Ｆ	

審査請求　未請求　請求項の数31　ＯＬ　（全 34 頁）　最終頁に続く

(21)出願番号　特願2000－42696(P2000－42696)

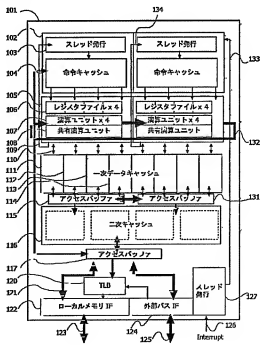
(22)出願日　平成12年２月21日(2000.2.21)

(71)出願人　597148312
進藤　苓介
広島県広島市西区己斐大迫３丁目20番５号
(72)発明者　進藤　苓介
広島市西区己斐大迫３丁目20番１号

最終頁に続く

(54)【発明の名称】　マルチスレッドを利用するパイプライン並列プロセッサ

(57)【要約】
【課題】マルチスレッドプログラムを利用するプロセッサにおいて、一列に並んだ演算ユニットで複数のスレッドを順に動作させる方式において、メモリ配線を短縮して周波数性能と並列性能を両立させる。さらに並列処理性能を阻害するノード間データ転送と、同期による待ちあわせを解決する。
【解決手段】特許広報平９－２８７６６２に記載されたプロセッサに、データを格納するキャッシュを複数搭載し、それぞれのキャッシュを数個の演算実行ユニットと結合する。キャッシュの内容はスレッドの進行にあわせて転送して複製する。キャッシュの内容を転送しきれない場合は、１つのスレッドを単一の演算実行ユニットで実行する。さらに、仮想記憶機構とキャッシュの共有機構を用いて、指定したアドレスへのアクセスを検出してスレッドを再開させる。



【特許請求の範囲】

【請求項１】数値演算ユニット、レジスタファイル、命令メモリ、データキャッシュメモリを複数個内部に有し、複数のスレッドおよびプロセスを同時に利用する事を特徴とするプロセッサにおいて、レジスタファイルが持つ各スレッドごとのレジスタ値等の状態を、常に隣接する演算ユニットに伝達することを特徴とするプロセッサ（以下ＰＭＴ方式プロセッサと称する）において、プログラムカウンタ、スタックポインタ値、スレッド識別番号、プライオリティー値で構成されるスレッドの情報を示す値を複数格納するスレッド情報格納手段を有し、スレッド情報格納手段から１つのスレッドを選択して、命令メモリおよび演算ユニットにスレッドの情報を伝送するスレッド発行手段を有し、スレッド発行手段において、スレッドの持つプライオリティー値を比較し、最も優先度が大きいスレッドの情報を優先的に命令メモリおよび演算ユニットに伝達することを特徴とするプロセッサ。

【請求項２】請求項１の特徴を持つプロセッサにおいて、演算ユニットが実行する命令のプログラムカウンタ値を保存する手段を有し、次に新規に発行する候補のスレッドが同じ命令アドレスを利用するかどうかを比較し、前に実行したスレッドと命令が一致したスレッドを優先的に選択して出力するための手段を有することを特徴とするプロセッサ。

【請求項３】ＰＭＴ方式プロセッサにおいて、状態を伝達すべき隣接する演算ユニットが別のスレッドの処理を優先的に行うことを感知して、その時だけスレッドの状態を隣接演算ユニットに伝達せずに同一の演算ユニットで処理を行うことを特徴とするプロセッサ。

【請求項４】ＰＭＴ方式プロセッサにおいて、複数のスレッドがそれぞれ利用するレジスタの値を複数のレジスタバンクに同時に格納するレジスタファイルを有し、各レジスタバンクの内の１つを同時に利用し、スレッドの進行に応じてレジスタバンクの内容を隣接する別のレジスタファイルに転送することを特徴とするプロセッサ。

【請求項５】ＰＭＴ方式プロセッサにおいて、現在実行しているスレッドを中断し、待機状態のスレッドを実行する操作が必要な際に、実行しているレジスタファイルの値を演算ユニットに伝送する代わりに、レジスタファイルの別のレジスタバンクに格納されている待機状態のレジスタ値を演算ユニットに伝達し、別のスレッドの演算を即座に行うことを特徴とするプロセッサ。

【請求項６】請求項５の特徴を持つプロセッサにおいて、レジスタファイルにレジスタ状態が格納されていないスレッドを実行する際に限り、レジスタファイルの内容をスタックポインタ値の示すメモリから自動的に読み出すことを特徴とし、現在レジスタファイルに格納されていて利用されないスレッドの状態をスタックポインタ値の示すメモリに自動的に書き出すことを特徴とするプロセッサ。

【請求項７】ＰＭＴ方式プロセッサにおける、１つのスレッドが利用するレジスタの値をメモリに保存する特別な分岐命令において、分岐命令の時点のスレッドのレジスタ値をレジスタファイルに保持することを特徴とし、保存されたレジスタの値を読み込む特別な分岐命令において、レジスタファイルに保持されていたスレッドの状態を利用することを特徴とするプロセッサ。

【請求項８】ＰＭＴ方式プロセッサにおいて、複数のスレッド識別番号及びスタックポインタ値をまとめて格納することを特徴とするスレッド自動発行機構を有し、スレッド発行命令によってスレッドを発行する際に、格納されたスレッド識別番号及びスタックフレームを自動的に割り当てることを特徴とするＰＭＴ型プロセッサ。

【請求項９】請求項４に記載された特徴を持つプロセッサにおいて、１つのレジスタファイルが複数の演算ユニットで共有され、レジスタファイルが複数の演算ユニットから１つを選択してデータを伝送することを特徴とする転送手段を有し、レジスタファイルの内容を隣接するレジスタファイルに複数回に分けて転送することを特徴とするプロセッサ。

【請求項１０】ＰＭＴ方式プロセッサの演算ユニットにおいて、値の一部の演算を行う部分演算ユニットを複数個有し、それぞれの部分演算ユニット内部に、部分演算ユニットにおける結果値と完全な演算を行った場合の結果値とが一致しないことを検出するオーバーフロー検出手段を有し、さらに完全な演算を行うための１つの完全演算ユニットを複数の部分演算ユニットに接続し、部分演算ユニットのオーバーフロー検出手段の演算結果の不一致の検出によって、完全演算ユニットに部分演算ユニットで利用した値を転送して演算を再度行うことを特徴とするプロセッサ。

【請求項１１】ＰＭＴ方式プロセッサにおいて、分岐後のプログラムカウンタ値が演算結果によって動的に変更され、分岐後のプログラムカウンタ値が確率的に予測できる条件分岐命令において、分岐後に実行されると予測される命令を格納する命令キャッシュを有し、命令キャッシュに分岐の結果を判別するための情報を有し、実際に分岐が実行された際に予測した分岐結果との一致を確認し、不一致の場合はスレッドを中断してスレッド発行ユニットに正しい分岐結果を転送することを特徴とするプロセッサ。

【請求項１２】ＰＭＴ方式プロセッサにおいて、複数の演算ユニットを複数のブロックに分配し、ブロックごとに専属の一次キャッシュメモリを有し、ブロック内の演算ユニット全てと接続して、データアクセスを行うことを特徴とし、さらに１つ以上の二次キャッシュメモリを有し、複数の一次キャッシュメモリと接続して、互いにデータアクセスを行うことを特徴とするプロセッサ。

【請求項１３】ＰＭＴ方式プロセッサにおいて、スレッ

ドが書きこんだメモリ内容をスレッド自身がメモリから読み出して利用する際に、利用するメモリ内容を複数のキャッシュメモリの間で転送することを特徴とし、複数のキャッシュメモリ間の転送はスレッドの進行と同じ方向、速度で伝達することを特徴とし、スレッドの進行にデータの伝達が間に合わない場合はスレッドを停止させることを特徴とするプロセッサ。

【請求項１４】ＰＭＴ方式プロセッサにおいて、プロセッサ内部に１つ以上のキャッシュメモリを有し、個々のキャッシュメモリをさらに複数のメモリバンクに分割し、それぞれのメモリバンクへのアクセス数を制限することを特徴とし、同時にメモリバンクへのアクセスを行うことを特徴とし、さらに、複数のメモリバンクの選択のためにメモリアドレスを利用することを特徴とし、同じキャッシュへの複数のアクセスが存在した場合は、１つのアクセスだけを行い、他のアクセスを保持して後で行うことを特徴とするプロセッサ。

【請求項１５】請求項１２に記載された特徴を持つプロセッサにおいて、キャッシュメモリ内部に、キャッシュメモリの内容の共有状態を指定するためのディレクトリと呼ばれる情報を有し、個別のキャッシュメモリは、別のキャッシュメモリから内部のデータを読み出された場合に、データのコピーを持つキャッシュメモリを特定する情報をディレクトリに設定することを特徴とし、同時に、別のキャッシュメモリから取得したデータをキャッシュメモリに格納する際に、データのオリジナルを持つキャッシュメモリを特定する情報をディレクトリに設定することを特徴とし、キャッシュメモリへの書き込みの際に、ディレクトリの内容を利用して、同じアドレスのデータのコピーを持つキャッシュメモリにだけデータの書き込みを通知することを特徴とするプロセッサ。

【請求項１６】ＰＭＴ方式プロセッサにおいて、ある命令が利用するデータを別の命令が再度利用する際に、データを再利用する命令を実行する演算ユニットを特定するデータフロー予測情報を命令メモリに格納することを特徴とし、データフロー予測情報を持つ命令が実行されたときに、データフロー予測情報で指定された演算ユニットにデータをあらかじめ転送することを特徴とするプロセッサ。

【請求項１７】請求項１６の特徴を持つプロセッサにおいて、あるスレッドのデータキャッシュアクセスミスの際に、データの実体のあるデータキャッシュからデータを読み込むと同時に、読み出しを行ったデータキャッシュに要求元の演算ユニットを特定する値を転送し、読み出しを行ったデータキャッシュに対応する命令メモリに、演算ユニットを特定する値を含むデータフロー予測情報を書き込むことを特徴とするプロセッサ。

【請求項１８】命令キャッシュメモリを複数有するＰＭＴ方式プロセッサにおいて、あるスレッドが、次に実行すべき命令を検索するためにキャッシュメモリにアクセスを行い、命令が格納されている命令キャッシュメモリを下位のキャッシュのディレクトリ情報から特定し、前記命令キャッシュメモリに接続された演算ユニットにスレッドを移動することを特徴とし、複数のスレッドが同一の命令キャッシュメモリを利用することを特徴とするプロセッサ。

【請求項１９】ＰＭＴ方式プロセッサにおいて、キャッシュメモリのアドレスを仮想アドレスとすることで、キャッシュメモリ上にはないデータへのアクセスに限って仮想記憶機構にデータを伝送し、仮想アドレスを物理アドレスに変換して物理アドレスメモリに書き戻すことを特徴とするプロセッサ。

【請求項２０】ＰＭＴ方式プロセッサにおいて、アドレス値を入力して、格納されたアドレス値に対する特定のスレッドを生起することを特徴とするデータフロー同期検出ユニットを有し、キャッシからの読み込み要求に対して、データフロー同期検出ユニットが指定したアドレスとの一致を判定し、一致するアドレスを含む場合はキャシュに共有状態を示す値を設定することを特徴とするプロセッサ。

【請求項２１】請求項２０の特徴を持つプロセッサにおいて、データキャッシュ内部で共有状態に設定されているアドレスへのアクセスに対して、ディレクトリの示すユニットにアクセスを通知することで、最終的にデータフロー同期ユニットにアドレス値を伝達することを特徴とし、データフロー同期ユニットが伝達されたアドレス値に対応するスレッドを生起することを特徴とするプロセッサ。

【請求項２２】ＰＭＴ方式プロセッサにおいて、スレッドは同期命令の発行時に停止し、他のすべてのスレッドの、同期命令実行前に行われたストア命令のデータ転送を待ち、すべてのデータが自身のキャッシュに転送された時点でスレッドを再開することを特徴とするプロセッサ。

【請求項２３】請求項２１のプロセッサにおいて、特定アドレスへのアクセスを検出する命令の発行によって、自分のスレッドの状態をデータフロー同期ユニットに自動的に伝達し、データフロー同期ユニットにおける特定のアドレスへのアクセスの検出によって自分のスレッドを再開することを特徴とするＰＭＴ型プロセッサ。

【請求項２４】ＰＭＴ方式プロセッサにおいて、１つのグローバル仮想記憶機構と複数のローカル仮想記憶機構を有し、複数のローカル仮想記憶機構がグローバル仮想記憶の値の一部を有することを特徴とし、グローバル仮想記憶機構の値の改変に対して複数のローカル仮想記憶機構に対して改変を伝達することを特徴とするプロセッサ。

【請求項２５】ＰＭＴ方式プロセッサにおいて、内部のユニット間で伝達する制御信号を、伝送先を示すアドレス値とともにまとめたパケットを利用して転送すること

を特徴とし、複数の制御信号を入力して、複数の制御信号の中から伝送相手に応じて選択して出力するパケットルーターを複数有し、ある演算ユニットからの要求を、パケットに変換して複数のパケットルーターが中継し、目的のユニットに伝達することを特徴とし、1つのユニット間配線を複数の制御信号で共有することを特徴とするプロセッサ。

【請求項26】請求項25に記載された特徴を持つプロセッサにおいて、スレッドが特定のユニットに制御信号を発信して、伝達したユニットから制御信号を受信する制御パケットにおいて、制御パケットをスレッドの進行方向と同一方向のパケットルーターに対して伝達することを特徴とし、制御パケットの伝達がスレッドの進行に間に合わないことを検出した場合は、該当するスレッドを即座に停止させることを特徴とするパケットルーター。

【請求項27】請求項25に記載された特徴を持つプロセッサにおいて、特定の制御信号パケットの要求に対して、該当する回路ユニットは要求された内部状態を改変、あるいは読み出して、制御信号を送信したユニットに対して内部状態を転送することを特徴とするプロセッサ。

【請求項28】PMT方式プロセッサを複数個利用して連結するシステムを構築する際に、プロセッサ間の転送方向を固定として、プロセッサのスレッドの状態、データをそのまま別のPMT方式プロセッサに伝送し、システム全体でスレッドを巡回させることを特徴とするPMT方式プロセッサ。

【請求項29】請求項28に記載された特徴を持つプロセッサにおいて、直接連結されていないプロセッサ間で独自にデータ転送を行うためのショートカットパスを設け、遠距離のプロセッサ間の伝送にショートカットパスを用いることを特徴とするプロセッサ。

【請求項30】請求項25に記載された特徴を持つパケットルーターを有し、請求項27に記載された特徴を持つPMT方式プロセッサにおいて、複数のプロセッサの全てのユニットをアドレス値で一意に特定する手段を持ち、スレッドの発行する制御信号パケットを、制御信号パケットの転送先アドレス値に応じて、外部のプロセッサ内部の該当するユニットに伝達することを特徴とするプロセッサ。

【請求項31】請求項30に記載された特徴を持つプロセッサにおいて、それぞれのプロセッサが独自にメモリを接続することを特徴とし、各プロセッサが持つ仮想記憶機構の内部に、指定されたページが外部のプロセッサのデータのコピーを格納していることを示す共有情報を有することを特徴とし、プロセッサ内部からデータを読み込む際に、読み込みアドレスが仮想記憶機構によって共有状態を示す場合には、プロセッサ外にデータ読み込み要求を行うことを特徴とし、プロセッサ内部からデータを書きこむ際に、書きこみアドレスが仮想記憶機構によって共有状態を示す場合には、プロセッサ外にデータの書きこみを通知することを特徴とするプロセッサ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ソフトウェアによって動的に機能を変更できるプロセッサに属し、特にスレッドと呼ばれる単位で分割されたソフトウェアを利用するプロセッサに属する。

【従来例】（半導体技術の進化とマイクロプロセッサの性能向上）

【0002】半導体技術の進化により、ここ20年はトランジスタ、配線の微細化が常に同じペースで進んできた。DRAMのようにトランジスタ数がそのまま容量につながる素子では、単に微細化によって素子数が増えるだけで、微細化と同じペースで性能を向上できた。

【0003】ところが、マイクロプロセッサに代表される論理LSIに関しては、性能向上には2つの方法がある。1つは動作周波数の向上。そしてもう1つは動作周波数あたりの仕事量である。

【0004】まず、微細化によってトランジスタのスイッチング速度の向上し、前者の動作周波数の向上が可能になった。さらに、後者の動作周波数あたりの仕事量の増加は、利用できるトランジスタの増加により、規模の大きい高速レイテンシ回路、およびスーパースカラなどの並列方式の採用が可能になったことで実現できた。

【0005】これまでは、マイクロプロセッサはこの2つの要素によって飛躍的な性能向上を可能にした。しかし、この2つの要素が、特に後者が限界を迎えつつある。この限界を打破しなければ、今後のマイクロプロセッサの性能向上は見込めない。

【0006】（配線のリスクの相対的増加）

【0007】近年の半導体の微細化技術、プロセスの進歩により、トランジスタの動作速度は飛躍的に増大し、その大きさ、消費電力も飛躍的に減少した。これによって、少なくともトランジスタ単位では、従来では考えられない周波数の動作が可能になった。

【0008】しかし、配線の遅延時間はそれほど改善されてはいない。配線長は、トランジスタのサイズに比例して高速化するわけではない。さらに、微細化された分だけトランジスタの数を増やす場合は配線遅延はかえって増大する。この傾向は深刻に受け止められてきており、配線が最小となるユニット配置を行うことは常識となっている。配線自体のプロセスによる改善も行われている。多層配線やCu配線などがそれである。しかし、それだけでは拡大を続けるトランジスタと配線の速度差を埋めることはできない。

【0009】今後は、配線遅延の増加を押さえて動作周波数の向上比率を維持するためには、常に回路全体に最短配線するという考え方を改め、レイテンシ性能を低下

させてでも最短距離の配線で伝送することが必要となる。

【0010】(データ転送スループットとデータ転送レイテンシ)

【0011】データ転送性能の向上には、データ転送スループットの向上とデータ転送レイテンシの短縮の双方が必要になる。前者のデータの転送スループットの増加は比較的たやすい。それに対して、転送レイテンシは性能低下を押さえるのが精一杯で、数倍以上の改善は見込めない。

【0012】レイテンシ向上の方法としては、キャッシュ、プリフェッチなどによる確率的な方法があるが、それは回路規模を必要とする割にたいした性能向上を果たせない。演算能力と低速なメモリとのレイテンシの開きは拡大の一途をたどり、キャッシュミスにおけるペナルティーを相対的に増大させ、最終的には処理時間のほとんどすべてを占めることになる。ということは、なんらかの形でレイテンシを隠蔽することが必須になる。

【0013】そのために現在はアウトオブオーダースーパースカラ、ＶＬＩＷという方式が存在する。データのロードが終わっていなくても、データの必要のない命令を先に動作させるプロセッサである。だが、この方式は先に実行させることができる命令を発見する回路が巨大になりすぎ、周波数性能向上に限界がある。

【0014】よって、レイテンシの隠蔽は今後さらに重要になる。だが、アウトオブオーダースーパースカラやＶＬＩＷなどの命令レベル並列では、現在以上のレイテンシ隠蔽は不可能である。

【0015】(演算ユニットの使用頻度のばらつきと共有)

【0016】マイクロプロセッサには、加算、論理演算、シフト、分岐、ロードストア、乗算、除算、浮動小数点演算、ＳＩＭＤ型演算、ＳＩＭＤデータの入れ替え処理など、多くの処理が必要とされる。これらの動作の実現には、それぞれ専用の回路を設けるのが一番効率が良い。ところが、マイクロプロセッサはこれらの全てを同時に必要とするわけではない。稼動率が低いユニットも多く存在する。

【0017】このマイクロプロセッサを同時に複数使用する方式を、マルチプロセッサと呼ぶ。現在のマルチプロセッサでは、これらの演算ユニットが全て複数搭載される。ということは、全体としてはほとんど稼動していない回路が増加することになる。仮に、マルチプロセッサの間であまり使用されない演算ユニットを共有できれば、システム全体の回路の利用効率を高めることができ、本当に数の必要な演算ユニットを増やすことができる。

【0018】(消費電力の増大)

【0019】近年のマイクロプロセッサの動作周波数の向上によって、消費電力は飛躍的に増大した。その増大を抑制するために、動作電圧を低減させ、低い電圧で性能を維持するための回路技術が開発された。しかし、回路素子数、周波数性能はさらに向上を続けるものと考えられる。さらなる低消費電力の手段が必要になる。

【0020】ＣＭＯＳ回路は、信号のレベルが変化するときに電力を消費する。ということは、信号のレベルの変化の少ない回路がもっとも消費電力の低い回路となる。回路構成のレベルでは、演算ユニットやクロック信号制御など、信号変化を低減する手段が多く利用されている。しかし今後は、さらに上位のアーキテクチャにおいても、最小の電力で演算を行うための手段が必要になると考えられる。

【0021】回路的に考えると、同じ仕事を連続して行うことができれば、回路の状態の変動も最小限となり、動作する回路も最小限となる。そして、トランジスタ数あたりの性能が向上できれば、逆にいえば性能あたりの消費電力が低減できるということである。

【0022】(演算内容の巨大化、分散化)

【0023】前の演算の終了を待ち、その結果を利用して演算を行うことを、データ依存関係と呼ぶ。互いにデータ依存関係のある演算は原理的に同時実行ができず、並列化を阻害する最大の要因である。いかなる方式もこれを解消することはできない。

【0024】ソフトウェアの構造上、このデータ依存関係がもっとも大きいのは遠統した命令の近傍であり、現在のスーパースカラやＶＬＩＷに代表される、命令レベル並列の対象とされる部分である。すなわち、命令レベル並列はもっとも並列化しにくい部分をあえて並列化する方法であり、性能向上に限界が生じる。

【0025】一般的に仕事の単位をうまく分割できれば、分業が効率が良いのは言うまでもない。そして、巨大なソフトウェアでは、その動作内容が全て密接に結合し、全ての命令、データが同じ確率で利用されるということはありえない。現に、ソフトウェアは、オブジェクトと呼ばれる独立性の高い単位で分割できることは良く知られている。

【0026】(データスループットの爆発的な増大)

【0027】メディア処理は、巨大なデータ転送能力を要求し、キャッシュの内部で実行できない代表的な処理である。この処理の多くは巨大なデータ転送スループットを要求する。それに対して、メディア処理は全体としてはさせてレイテンシを要求しない。要求されるレイテンシはどんなに小さくても１ミリ秒程度がせいぜいである。レイテンシを犠牲にして並列処理を行うのにこれほど向いた用途はない。

【0028】局所的なレイテンシがそのまま総和される現在のプロセッサの方式では、プロセッサパスのレイテンシがそのまま加算され、全体の性能向上も頭うちになる。それに対して、レイテンシをなんらかの手段で隠蔽することができれば、メモリアクセスの並列化などの方

法によってスループットを確保することができる。そのためにマルチスレッドと呼ばれるソフトウェアモデルを導入して、レイテンシの悪化を防止する。スレッド単体のレイテンシが多少大きくてもメディア処理に要求されるレイテンシよりはるかに小さいため、結果的にメディア処理に要求される性能を全て満足することができる。

【0029】（演算の繰り返しの増加）

【0030】長時間動作するプログラムは、その全ての時間に渡ってまったく違う命令を実行することは考えられない。そのため、長い時間の動作の中では、何らかの形で同じコードを再利用して同じ動作を繰り返していることになる。

【0031】この傾向を利用することにより、同じ動作を行う部分を同時にまとめて実行することで、同じ動作で共有される命令メモリ、データメモリなどの資源を共有することができる。しかも、まったく同じ動作を時間的にわずかにずらして実行することにより、同じ資源を同時に利用することも簡単に防ぐことができる。

【0032】（ＩＰユニットの内蔵と、それを結合する性能の要求）

【0033】汎用プロセッサは、３２ビットなどの桁の多い数値演算や、大容量メモリ全域を利用した処理、動的に変わる処理に関しては他の手段では実現不可能な性能を発揮できる。しかし、少数の複雑なビット処理演算に関しては依然として弱く、目的に応じて最適化された回路の方が常に性能が上である。ということは、システム全体の性能向上のためには、依然として良く利用されるビット演算を担う回路、ＩＰ回路を内蔵することが望ましい。

【0034】ところが、ＩＰ回路は、その前後の動作がなければ十分な性能が発揮できない。ＩＰ回路同士を直接連結すると、その回路の動作の種類を制限することになる。プログラマブルでかつ高速なアプリケーションの動作を実現するためには、複数の最小限度のＩＰ回路と、ＩＰ間のデータの中継を行う十分な演算処理能力が最良の組みあわせである。

【0035】（スーパースカラ、ＶＬＩＷ方式）

【0036】スーパースカラ方式、ＶＬＩＷ方式は、命令レベル並列とよばれ、同時に複数の命令を実行することで、性能を向上させることを狙った方式である。

【0037】まず、スーパースカラ方式は、複数の命令の組みあわせを自動的に抽出してくれる方式である。ところが、自動的に抽出できる命令の範囲、命令ウィンドウは限定されており、特に、条件分岐命令の後に実行される命令の抽出が非常に難しい。そのため、プログラム全体の並列性を生かすことができず、隣接した数個の命令を実行するのがせいぜいである。

【0038】図２に、従来のプロセッサ例としてＶＬＩＷ方式のプロセッサの構造模式図を示す。ＶＬＩＷ方式は、この命令の抽出の手間をコンパイラに任せ、並列可能な命令を明示して命令メモリに格納する方法である。しかし、並列化の対象となるのはプログラム内部で隣接した数個の命令であることには変わりない。

【0039】２０１は複数の命令を同時に格納する命令キャッシュである。命令発行ユニット２０８は、命令キャッシュ２０１から同時に複数の命令を読み込み、送られた命令を実行できる演算ユニットにそれぞれ命令を分配する。演算ユニット２０２、演算ユニット２０３、分岐ユニット２０４、ロードストアユニット２０５は、同時に独立した動作ができる。演算ユニット２０２、演算ユニット２０３は、共有レジスタファイル２０６から複数の値を取り出して演算を行い、結果をレジスタファイル２０６に返す。分岐ユニット２０４は、命令キャッシュ２０１に対してＰＣアドレスを変更させる。ロードストアユニット２０５は、データキャッシュ２０７からレジスタファイル２０６にデータを読み込む。あるいは逆に、レジスタファイル２０６の値をデータキャッシュ２０７に転送する。

【0040】（マルチプロセッサ方式）

【0041】図３に、従来例としてマルチプロセッサ方式を示す。マルチプロセッサ方式は、既存のパイプライン、スーパースカラ、ＶＬＩＷのいずれかの方式で作成されたプロセッサを複数接続して利用する方法である。飽和しつつある命令レベル並列を補うために用いられる。

【0042】そのために、ソフトウェアをプロセス、あるいはスレッドとよばれる独立した単位に分割して、それぞれのプロセッサに割り当てる。それぞれのプロセッサはそれぞれ独立したスレッドを実行することで、命令レベル並列に対して演算ユニット間の通信を抑制することができる。

【0043】図３にマルチプロセッサの構造を示す。プロセッサ３０１、３０２、３０３、３０４は、共有バス３０５に接続される。プロセッサ３０６、３０７、３０８も同様に共有バス３０９に接続される。共有バス３０５には二次キャッシュ３１０が接続され、プロセッサ３０１のメモリは基本的には二次キャッシュ３１０から取得する。２つの二次キャッシュ３１０、３１１は、共有メモリバス３１２に接続され、二次キャッシュとメインメモリ３１３の内容を同一にする。

【0044】プロセッサ３０１～３０４、３０６～３０８は、それぞれ独自に命令動作を行い、命令、データをメインメモリ３１３からキャッシュを介して取得する。他のプロセッサと同一アドレスのデータを共有しない限り、プロセッサ間通信は行われない。

【0045】これらのプロセッサ、二次キャッシュ３１０，３１１は、半導体のチップに全て搭載することが可能である。半導体チップの微細化によって、同じコストでもより多くの回路の搭載が可能になったため、複数のプロセッサを１つのチップに搭載することで、コストに

対する性能を向上させることになる。

【0046】（従来のPMT方式）

【0047】図4に、命令レベル方式、およびマルチプロセッサ方式の欠点を解消するための従来の方式を示す。以下、この方式をPMT方式と呼称する。PMT方式についての詳細は特許出願平9－287662に記載されている。

【0048】このPMT方式は、前述のマルチプロセッサ方式で利用されるプロセス、スレッドをほぼそのまま用いる。そして、演算ユニット間の通信を最小限にすることにより、演算ユニットの増加に対して周波数性能の低下を抑制し、動作周波数を維持しつつ大量の演算ユニットの搭載を可能にし、飛躍的な性能向上を可能にする。さらに、演算ユニットなどの回路を可能な限り共有することによって、最小の回路規模で最大の並列規模を達成できる。

【発明が解決しようとする課題】

【0049】（VLIW方式の欠点）

【0050】VLIW方式の欠点を示す。まず、命令レベル並列は、プログラムの局所的な領域だけで実行できる命令を選択する方式である。理由は、プログラムはその場の演算結果によって命令の流れが頻繁に変更されるため、演算が終了するまで次に実行すべき命令を特定することはできない。それをある程度克服するために分岐予測と呼ばれる機構があるが、それでも複数の分岐の先を予測することは難しい。そのため、命令キャッシュ301の幅を広げても、同時に実行できる命令をプログラムから大量に選択できないため、性能向上率が飽和する。

【0051】さらに、複数のデータ依存関係が発生するということは、それらの命令の間のデータの自由な転送が必要になるということである。一般的に、命令実行ユニットのN倍の増加に対して、実行ユニット間の配線の遅延時間はN倍以上、回路規模はNの二乗の規模で増加する。そのため、命令実行を増やしても、それ以上に周波数性能が低下するというデメリットが生じる。

【0052】以上の理由によってVLIW方式は性能向上に限界がある。

【0053】そのため、命令発行ユニット208の幅を広げるのはあきらめて、複数の明示的に独立したスレッドを1つのプロセッサで同時に実行するのも必要と考えられるようになった。そのため、小規模なVLIWを複数搭載し、個々のVLIWで個別のスレッドをそれぞれ動作させるという方法が考案されている。ところがそれでは、次に述べるマルチプロセッサ方式の問題が発生する。

【0054】（マルチプロセッサ方式の欠点）

【0055】次に、マルチプロセッサ方式の4つの欠点を示す。

【0056】まず、マルチプロセッサでは、負荷の高いプロセッサから負荷の低いプロセッサへプロセス、あるいはスレッドを移すのに非常に時間がかかる（以下、このプロセス、スレッドの移動をプロセス移住、スレッド移住と呼ぶ）。

【0057】次に、マルチプロセッサにはプロセッサ間通信が必要になる。複数のプロセス、スレッドがまったく独立したデータを利用することはまれであるためである。ところが、1つのデータを全てのプロセッサが利用すると、データ通信の量はプロセッサの数にほぼ比例して増加する。そして、通信の量が増えるということは、単体のプロセッサから見てもメモリのアクセスが通信、同期によって制限されることになり、単体のプロセッサにおいても、システム全体においても性能が飽和する。

【0058】次の問題は、プロセッサ間の同期である。あるプロセスがほかのプロセスの特定の処理を待つために停止し、別のプロセスからの処理終了の伝達によって再開するのが同期である。このための最も原始的な手法は、待ち状態のプロセスが定期的に別のプロセスの状態を監視することである（スピンロックと呼ばれる）。しかし、これでは待ち状態のプロセスがプロセッサ、メモリバスなどの資源を占有するために非常に効率が悪い。そのために、OSレベルのソフトウェアで同期処理を管理する方法などがあるが、そのためのソフトウェア処理が大規模な並列における性能向上を阻害するという問題がある。

【0059】最後に、マルチプロセッサは、メモリ、複数の演算ユニットをすべて搭載するプロセッサを、さらに複数搭載する。そのため、それぞれの演算ユニット、メモリの稼動率にもかかわらず、すべてのコピーがプロセッサの数だけ搭載されることになる。そのため、回路規模の点で無駄が多い。

【0060】（従来のPMT方式の欠点）

【0061】PMT方式は、以上で述べた、VLIWに代表される命令レベル方式の性能の限界、およびマルチプロセッサ方式の回路規模的な欠点を解消するための方式である。

【0062】まず、複数のスレッドを常に全てのユニットで巡回させることで、スレッド発行ユニットを演算ユニット間で共有できる。さらに、全てのスレッドを空いた演算ユニットに対して即座に発行することができ、スレッドを中断した場合も、スレッドの移住を行わなくてもその場で再開が可能である。これによって、レイテンシを隠蔽するためのスレッドの切り替えを高速に行うことができる。

【0063】複数のスレッドを動作させる際には、データキャッシュの内容を共有することが多い。そのため、スレッド間で同じデータキャッシュを共有することで、全てのキャッシュへ同じデータを転送する必要が無くなり、ブロードキャスト型のデータの転送を最小限にすることができる。

【0064】同じ種類のスレッドは、同じ命令、データメモリ、演算ユニットを利用する傾向が強い。この性質を利用して、1つの命令キャッシュ、データキャッシュ、特殊演算ユニットを複数のスレッドから共有させることで回路を削減することができる。

【0065】だが、従来例に挙げた図4のPMT方式には、以下の欠点がある。

【0066】まず、コンテキストスイッチのために、メモリにレジスタの値の退避が常に必要になる。キャッシュミスのように、もとの演算ユニットでスレッドを再開できるような処理では、演算ユニットにレジスタを保持しておけば、スレッドの移住は必要ない。そのために、複数のスレッドを同時に管理するレジスタファイルが必要になる。

【0067】次に、分岐命令ごとにコンテキストスイッチが必要になる。理由は、命令アドレスに対して、実行される演算ユニットが常に決定されているために、命令アドレスが昇順に実行されない場合はスレッドの移動が必要になるためである。分岐命令はソフトウェア全体で4分の1を占めるともいわれるため、このようなスレッドの移動は大きく性能を低下させる。ソフトウェアのインライン展開によってある程度分岐を減少させることは可能であるが、汎用的なソフトウェアで性能が出る構造が望ましい。

【0068】次に、命令アドレスによって実行される演算ユニットが決定されるため、命令の配置によっては演算ユニットの稼動率にバラ付きが生じる。同じようにソフトウェアのインライン展開でうまく大半の演算ユニットを利用することはできるが、汎用的なソフトウェアで負荷分散が出来る構造が理想的である。

【0069】従来のPMT方式では、キャッシュ間でデータのコピーを持たせないために、全ての実行ユニットが全てのキャッシュメモリと接続するように配線させる必要がある。そのため、実行ユニットのN倍の増大にしたがってNの二乗で規模が増大する。配線遅延が深刻化する現在では、このような配線は確実に周波数性能を低下させる。ところが、性能向上の為には実行ユニットを増加させることが不可欠である。そのため、キャッシュのコピーを各実行ユニットに持たせる必要があり、キャッシュ間の内容の整合性を取るハードウェアを実装する必要がある。

【0070】従来のPMT方式では、キャッシュのコピーを一切行わないため、全てのキャッシュのアクセスは順序が入れ変わることはない。ところが、キャッシュのコピーを持たせる構造にすると、キャッシュのアクセス順序を保持できなくなる。そのため、新たなハードウェアによる同期機構によって、キャッシュのアクセス順序を保証する必要がある。

【0071】最後に、全てのスレッドは全ての資源に無制限にアクセス可能であり、同時に独立したプロセスを動作させることができない。そのためには仮想記憶機構によるプロセス間保護の実装が必要である。ところが、キャッシュメモリを分散させると、仮想記憶機構はキャッシュメモリの数だけ必要になる。キャッシュメモリは複数のプロセスが混在するため、単体の仮想記憶の容量も増大する。更に、仮想記憶機構を分散させると仮想記憶の規模が膨大なものになる。

【0072】以上が従来のPMT方式の欠点である。PMT方式の持つ長所を維持しつつ、これらの欠点を解消するのが本発明の目的である。

【課題を解決するための手段】

【作用】

【0073】(コンテキストスイッチ)

【0074】本発明のプロセッサはマルチスレッドを利用する。マルチスレッドは大規模なレイテンシを隠蔽する唯一の方法と言ってよい。このマルチスレッドの管理は、従来のマルチプロセッサなどではOSの仕事となっているが、それがスレッドの数に比例して処理時間を増大させて、マルチスレッドの長所をほとんど発揮できない要因となっている。ハードウェアで極力マルチスレッド動作を実現するのが望ましい。

【0075】図16にマルチプロセッサにおけるマルチスレッドの実行例を示す。スレッドAからスレッドBへの切り替えを行うスケジューリングは、常にプロセッサの資源を消費する。さらに、キャッシュミスの期間には、他のスレッドの動作ができず、各プロセッサはアイドリング状態となる。

【0076】図17に、本発明のプロセッサにおけるマルチスレッドの実行例を示す。本発明のプロセッサでは、複数のスレッドがストールしない限り、スケジューリングを全てハードウェアで行うため、常に演算ユニットを実際の動作に利用することができる。キャッシュミスの場合も、別のスレッドがかわりに動作することができる。キャッシュの入れ替え動作が終了した後は、別の任意のスレッドのストールによって、スレッドを再開することができる。

【0077】結論として、本発明のプロセッサは、マルチプロセッサ方式に対してコンテキストスイッチ、スケジューリングの時間が不要である。さらに、本発明のプロセッサはあらゆるスレッドの待ち時間に他のスレッドが動作可能であり、どんなに並列度を上げても演算資源をほぼ常時利用することができる。これは、現在の命令レベル並列では、数並列程度でも演算資源の利用率が半分以下であるのと対照的である。

【0078】複数のスレッドを同時に動かす際には、待ち状態のスレッドの中から演算能力に相当する数のスレッドを選択することが必要になる。スレッドには、例外や割り込み要求の応答など、即座に実行を要求されるものと、比較的実行遅延が許されるものとが混在している。このため、スレッドの優先順位を設け、それを自動

的に選択する機構が必要になる。

【0079】本発明のプロセッサにおけるスレッドは16段階のプライオリティーを有する。スレッド発行ユニットは、実行待機状態のスレッドを格納し、スレッドのプライオリティーをハードウェアで判定して選択して、同時に1つのスレッドを発行する。また、既存のスレッドよりもバッファ上のスレッドの優先度が高い場合は、無条件で既存のスレッドを休止して新規のスレッドを発行する。プライオリティーが同一の場合はとくに優先制御、負荷分散制御を行う必要はない。

【0080】なお、実行ユニットの稼動率が高く、新規のスレッドを発行できない場合は、隣接するスレッド発行ユニットに順にスレッド状態を転送する。

【0081】スレッド発行ユニットが発行すべきスレッドを選択する際に、前に実行したスレッドと共通の命令を利用するものが理想的である。理由は、命令が同一であれば利用するデータも同じである確率が高いこと。そして、命令などの状態が等しければ、制御回路などの状態の変更が最小限となり、状態信号が変化しなければCMOS回路の特質上消費電力が最小となるためである。

【0082】そのために、前に発行したスレッドの命令アドレスを控えておく。そして、次に発行するスレッドの命令アドレスと、控えておいた前のスレッドの命令アドレスを比較し、同一であればスレッドを即座に発行する。アドレスが同一でない場合は、今のスレッドとプライオリティーが同一以上のスレッドがない場合に限り用意したスレッドを発行する。

【0083】PMT方式では、そのままではスレッドのライン間の移動によって演算ユニットの間で負荷のばらつきが生じる。そのため、ある演算ユニットは負荷が極端に高く、どうしてもほかのスレッドの要求を受け付けられない状態が発生する。そういう場合は、空いた1つの演算ユニットを有効活用するために、その演算ユニットを単一プロセッサとみなしてスレッドの実行を行う（今後、この動作を局所SMP実行モードと称する）。こうして、PMT方式とSMP方式を混在させて、スレッドが充填されない演算ユニットを最大限に活用する。プライオリティーの高い別のスレッドの要求によって、局所SMP実行モードは解除される。

【0084】スレッド発行ユニットが4つの演算ユニットで共有される場合は、局所SMP実行モードは4つの演算ユニットを順に利用して行う。この場合、4つのスレッドが同時に動作することになるが、相互の演算ユニット間のレジスタ、データ転送は不要である。

【0085】コンテキストスイッチを高速化するために、従来のPMT方式にあったレジスタのメモリへの待避の必要性をなくす。そのために、レジスタファイルには複数のスレッドの情報を共存させ、そのうちの1つだけを利用する。コンテキストスイッチは、利用するレジスタファイルのバンクを切り替えるだけで良く、即座にスレッドを切り替えることができる。

【0086】PMT方式では、スレッドは基本的には一定方向に移動する。しかし、命令、データの共有を実現するためには、すでに命令が保持してある演算ユニットにスレッドを移すことが望ましい。あるいは、すでに負荷の高い演算ユニットに到達したときは、負荷の低いラインに移動する必要がある。そのために、演算ユニット間でスレッドを移動させる、スレッド移住機構を設ける。スレッド移住は以下の手順で行う。

【0087】（1）実行ユニットからストール要求。同時にレジスタバンクを別のスレッドに切り替える。

【0088】（2）スレッド発行ユニットは待機してあるスレッドを供給。

【0089】（3）データキャッシュにレジスタの内容を退避。直接二次キャッシュに対して送られる。

【0090】（4）目的のノードにスレッド情報転送。

【0091】（5）データキャッシュ階層を通って、目的のノードに近い二次キャッシュからレジスタの読み込みを行う。データキャッシュ間の転送は、後述のキャッシュコヒーレンシ機構を用いる。

【0092】なお、本発明では、負荷分散のためのスレッドの移住は基本的には不要である。待ち状態のスレッドは一定場所にとどまっていれば、いつかは他のスレッドが使用していない空いたパイプラインが流れてくるためである。

【0093】図22に、スレッド移住における動作を示す。横軸は演算ユニットの列であり、縦軸は時間経過である。斜線が個別のスレッドの実行を示す。

【0094】7番の演算ユニットへのスレッドの移住によって、7番から10番の演算ユニットはメモリからレジスタを読み込む。11番の演算ユニットから実際のスレッドが再開される。

【0095】プライオリティーの低いスレッドは、7番の演算ユニットがプライオリティーの高い別のスレッドによって占有されたことを検出して、2番の演算ユニットの時点でスレッドを停止させる。3番から6番の演算ユニットではレジスタ状態をメモリに待避する。7番の演算ユニットから別のスレッドの移住が始まる。

【0096】一般的にサブルーチンコールでは、それまでのレジスタをスタックに保持して、リターンの直前に退避したレジスタを読み込む操作が必要になる。本発明のプロセッサでは、サブルーチンコールはレジスタを隣接転送する際に、元のレジスタを破壊せずに、サブルーチンコールを実行した演算ユニットのレジスタバンクに保持しておくだけで実現できる。そしてリターンはその保持されていたレジスタバンクを再利用して、帰り値を示す1つのレジスタだけを代入すれば良い。

【0097】図20に、サブルーチンコールの動作例を示す。CALL命令がサブルーチンへの分岐、RET命令がサブルーチン終了を示す命令である。

【0098】CALL命令のように、元の命令アドレスに戻り、元のスタックの値を利用する処理においては、CALL命令の位置にレジスタ値を残しておくだけで良い。レジスタはコール先の命令にも複製されて継承される。

【0099】RET命令の実行によって、帰り値だけがCALL命令に送られる。それ以外のレジスタは、元のレジスタの値をそのまま利用すればよい。

【0100】保持してあるレジスタバンクをほかのスレッドが利用するときは、前述のスレッド移住機構におけるレジスタ同期機構によって、自動的にメモリへの退避が行われる。

【0101】割り込みユニットやTLBは、蓄積されたスレッドIDをスレッド発行ユニットに伝達し、指定されたスレッドを動作させることができる。

【0102】そして、TLBからのスレッド生起は、ページフォルトなどのTLB例外によるコンテキストスイッチを高速化するとともに、OSカーネルサービスの並列化を実現する。

【0103】本発明のプロセッサは大量のスレッドを利用する。そのためには、現在進行しているスレッドの演算能力を極力利用せずに、大量のスレッドを発行しなくてはならない。そのためめに、スレッドが必要なスレッドID、スタックなどの情報（スレッド構造体と呼ぶ）はハードウェアで管理して、スレッドの生成によって自動的に転送する。実装としては、まとめてスレッドIDとスタックポインタを格納するバッファだけを設ける。バッファの内容の管理はまとめてソフトウェアで行う。

【0104】スレッドを発行する場合は、スレッドバッファから空き状態のスレッド構造体を要求する。スレッドバッファにスレッド構造体が無い場合は、現在のスタックポインタをそのまま返し、これ以上マルチスレッドで実行できないことをプログラムに通達する。

【0105】こうして、スレッド発行命令は新規のスレッド構造体を取得する命令だけで済むようになり、スレッド発行におけるソフトウェアオーバーヘッドを削減できる。

【0106】（演算パイプライン）

【0107】本発明のプロセッサは、レジスタファイルを隣接する複数の演算ユニットで共有する。4つの演算ユニットでレジスタファイルを共有する場合は、4つのレジスタファイルと4つの演算ユニットとの間で自由にアクセスするためのクロスバ接続バスを設ける。

【0108】こうして、従来のPMT方式が常にすべてのレジスタの値を隣接するユニットに転送を必要としたのに対して、隣接するレジスタファイルへの転送を数クロックに1回に抑制することができる。

【0109】レジスタファイルには複数のスレッドの情報が混在するが、一度に送るのは1つのスレッドのさらに4分の1の内容で十分となり、実行ユニット全体でも、1つのスレッド分のレジスタ転送だけで良い。

【0110】なお、同一の命令を利用するスレッドを連続して動作させている場合は、転送する信号の変化はスレッド間のレジスタ値の違いだけとなる。この違いだけがCMOS回路における消費電力となる。

【0111】本発明のプロセッサは浮動小数点演算ユニットを搭載することができるが、このユニットは整数演算に比べてレイテンシが大きくなるという特徴がある。その間、依存関係のない別の整数演算命令を実行することで、浮動小数点演算のレイテンシを隠蔽できる。

【0112】同一の命令を用いるスレッドを連続動作させる場合は、長レイテンシ演算も1つのユニットを使いまわすことになる。この場合は、1クロック分の演算が終了した時点で、隣接する別の長レイテンシ演算ユニットに中間値を渡し、並行して演算を行う。こうして、長レイテンシ演算のスループットを向上させる。

【0113】本発明のプロセッサは、一般的なパイプラインプロセッサと同じく、パイプラインを停止するパイプラインストールを発生する機能を有する。ただし、パイプラインプロセッサと違う点は、ストールする対象が単独のスレッドに限られ、ほとんどの種類のストールの間に待ち状態の別のスレッドを再開できる点である。

【0114】パイプラインストールは、一般的にはあるスレッドの要求する演算ユニット、あるいは転送バスなどの資源を取得できなかった場合に発生する。そして、ストール状態のスレッドは、その原因が解決された時点で、プライオリティーの低い別のスレッドの動作を中断することができる。

【0115】パイプラインストールは、すでに実行してしまった演算内容を1，2命令分キャンセルする必要がある。たとえばロード命令に対して、ロード命令が利用するキャッシュへのインバリッドの伝達が間に合わなかった場合、そのロード命令を無効にする必要がある。

【0116】図21は、パイプラインストールの動作例である。スレッドAのEXステージの実行が失敗して、別のEX'ステージによる実行が必要になる。スレッドAの待避したパイプラインスロットには、前にパイプラインストールを起こした別のスレッドが入り込み、結果を格納する。

【0117】EX'の具体的な動作は64ビット演算や浮動小数点除算などである。演算自体は数クロックで終了し、再開待ち状態となる。スレッドEのパイプラインストールによって、スレッドEのかわりにスレッドAが入り込み、スレッドAの命令を終了させる。

【0118】パイプラインストールごとにスレッドを切り替えることによって、パイプラインを間断無く動作させることができる。ただし、パイプラインストールが発生した命令が、前にパイプラインストールが発生した命令より後である場合は、パイプラインに空きが生じる。ただしその幅は最大4クロックである。しかも、同一命令

を利用するスレッドを連続動作させる場合は、パイプラインストールを起こす命令も同一である確率が高いため、大きなペナルティーにはならない。

【0119】(ディレクトリ方式階層キャッシュ)

【0120】大量の演算ユニットを搭載するには、それに対応するだけのデータ転送能力が必要になる。ところが、1つのメモリから大量のデータを供給することは不可能である。何らかの形でメモリを分散するしかない。ところが、本発明の方式では、全ての演算ユニットから全てのメモリを高速に参照する必要がある。そのために、分散したメモリの間でコピーを持つ必要がある。

【0121】分散されたメモリは、本来のメモリのコピーを自動的に格納するキャッシュの形態を取る。このとき、キャッシュ間で同じデータのコピーを持つ場合は、あるキャッシュへの書きこみを、別のキャッシュへと転送しなくてはならない。このコピー間のデータの整合性を取る機構を、キャッシュコヒーレンシ機構と呼ぶ。

【0122】ところが、キャッシュの数が増大すると、キャッシュの間の転送量も増大し、配線の量、遅延時間も増大する。キャッシュ間で接続されるバス信号の数を最小限度にし、かつキャッシュ間の転送スループットを確保するために、階層型キャッシュ構造を取る。

【0123】演算ユニットには専用の一次キャッシュが接続され、複数の一次キャッシュに対して1つの二次キャッシュが接続される。遠距離の一次キャッシュへの転送に関しては、二次キャッシュを介して転送される。一次キャッシュと二次キャッシュの間のデータパスの接続はクロスバ接続であり、転送スループットを確保する。ただし、クロスバ接続の組みあわせは4つ程度に限定し、配線規模の増大を防ぐ。

【0124】本発明のプロセッサにおいては、隣接しないキャッシュ間の転送は即座には行われない。二次キャッシュにいったん格納されてから伝達される。

【0125】ここで、データの書きこみを行ったスレッド自身が同じデータを読み込む場合を考える。キャッシュ間の転送が間に合わなければ、自分自身のデータも読めないことになる。しかし、キャッシュ間の転送はスレッドの進行に間に合えば良いため、多少のレイテンシの遅れは許される。

【0126】特に、二次キャッシュ間の長距離配線、大容量の二次キャッシュは動作レイテンシが遅くなる傾向がある。ところが、二次キャッシュアクセスを長距離の演算ユニットの間の転送に用いれば、その距離の間のスレッドの進行に間に合えば良いため、キャッシュ動作レイテンシを隠蔽できる。

【0127】異なるスレッド間では、スレッド間の同期を行わない限りデータの即座な転送を保証する必要はない。同期を行う場合は後述する。

【0128】二次キャッシュは複数の一次キャッシュ、そして隣接する二次キャッシュ、三次キャッシュからの要求をすべて受理することになる。これらの転送スループットは膨大なものとなり、同時に複数の要求を受理しなくてはならない。しかし、同時に複数の要求を完全に受理できる、マルチポートのメモリ回路は規模も大きく、速度も遅い傾向がある。

【0129】そのために、一次キャッシュは複数のロードストアユニットに接続する。逆に1つのロードストアユニットからは、複数の一次キャッシュをアドレスによって選択する。二次以上のキャッシュは複数のバンクに分割し、同様にアクセスするアドレスによってバンクを選択する。同時に同じバンクへのアクセスが重なった場合は、片方のアクセスを停止させる必要がある。ただし、本発明のプロセッサのキャッシュ間のデータ伝送は、スレッドの進行に間に合えば良いため、多少の衝突による遅れは許容される。この機構によって、確率的に多ポートのキャッシュに近いスループットを確保できる。

【0130】データのコピーを持つ別のキャッシュを特定するためには、バススヌープ方式とディレクトリ方式の2つの方法がある。バススヌープ方式は、共有の可能性のあるデータを共通のバスに出力し、全てのプロセッサが共有状態かどうかを判定する方式である。このバススヌープ方式の利点は、共有判定のための外部回路が単純であること、複数のプロセッサへの同時転送が可能であることである。欠点は、すべての外部メモリアクセスがメモリバスを占有して、全体の転送スループット性能を低下させる点と、すべてのプロセッサが自身のキャッシュをアクセスしてコピーを持つかどうかのチェックを行う必要があるという点である。市販されているスーパースカラ型マイクロプロセッサはバススヌープ方式を採用することが多い。

【0131】これに対して本発明のプロセッサは、データの転送スループットが重要であり、かつデータの転送相手を限定する必要がある。そのため、共有するプロセッサを明示的に指定するディレクトリ方式を採用する。ディレクトリ方式は、キャッシュの内部にデータの共有相手を特定するための情報を持つ。

【0132】図23にディレクトリ方式階層キャッシュのロードにおける挙動を示す。演算ユニットからのロードの場合、一次キャッシュ内部にデータがない場合に限り、二次キャッシュから一次キャッシュに向けてデータを転送し、二次キャッシュに共有状態を設定する。すでに二次キャッシュのデータが共有状態となっている場合は、ディレクトリビットの示す一次キャッシュに対して共有状態を設定する。

【0133】図24は、ディレクトリ方式階層キャッシュのストアにおける挙動である。一次キャッシュへの書きこみの際に、一次キャッシュのエントリが共有状態となっている場合は二次キャッシュに書きこみを通達する。二次キャッシュはディレクトリビットの示す共有相

手に対してのみ、直接キャッシュエントリの無効化（インバリッド）を通知する。

【0134】ディレクトリの指定により、一次キャッシュには確実にデータのコピーがあることが判明するため、一次キャッシュのタグの比較を行う必要なく、直接書き込みを行うことができる。ただし、セットアソシアティブキャッシュの場合は、ディレクトリビットは単体のキャッシュ内部のどのバンクにデータが格納されているかを指定する必要がある。

【0135】同じ命令を利用するスレッドは、たとえアクセスするアドレスが異なっても命令間のデータの流れは等しい場合が多い。レジスタの場合は明示的にプログラムで示されるが、メモリに対しても同じことが言える。特に、スタック、ヒープなどを利用する命令では、アドレスは異なっても命令間のデータの流れは等しい場合が多い。

【0136】本発明のプロセッサでは、同一スレッド内部でのキャッシュミスを極力減らすために、たとえキャッシュの共有情報がなくても、ストアされたデータを可能な限り事前にロード命令に渡す必要がある。

【0137】そのために、命令アドレスに対してデータフロー予測情報と呼ぶ情報を設ける。データフロー予測情報がマークされた命令は、ロード、ストア命令で使用したのデータを自動的に次のロード命令に伝達する。そのために、データフロー予測情報には、伝達先のキャッシュを特定する値が格納される。データフロー予測情報は、命令によって明示的に組み込むことも、自動的にプロセッサが書きこむことも可能である。

【0138】データフロー予測情報は、プログラムで明示的に記述するのが簡単だが、既存のソフトウェアとの互換性、そして条件によってデータアドレスが動的に変更される場合に対処するために、ハードウェアで自動的に設定するのが望ましい。

【0139】図19に、データフロー予測情報の書きこみ動作を示す。ロードストアユニット1907における、最初の命令実行でキャッシュミスを起こした命令は、キャッシュの共有状態からデータの実体の位置を知る。そして、データの実体のあるキャッシュ1904からデータを取得すると同時に、データの実体を持つキャッシュ1904に向けて、自分の演算ユニット1906を示す値を送る。こうして、データの実体のあるキャッシュ1904は命令キャッシュ1901にデータフロー予測情報を書きこむ。

【0140】（命令キャッシュ）

【0141】本発明のプロセッサでは、複数のスレッドが同じ命令を利用し、同じ命令は同じ演算ユニット、データキャッシュを利用するのが望ましい。そのためには、発行されたスレッドがプログラムカウンタから命令キャッシュの場所を特定し、自由にスレッドを移動させることが必要になる。

【0142】図18は、分岐によるスレッド移住の方法を示す模式図である。一次キャッシュ1803などに格納された命令は、二次キャッシュ1801に格納されたディレクトリに共有状態を設定する。命令キャッシュ1808の命令キャッシュミスか、分岐命令1806による要求によって二次キャッシュ1801にアクセスしたスレッドは、ディレクトリビットによって該当する命令が格納されている命令キャッシュ1802の位置を知り、その命令キャッシュに向けてスレッドを移住させる。

【0143】どの命令キャッシュにも命令が格納されていない場合は、スレッドの情報を動かさずに、分岐命令の直後、あるいはキャッシュミスを起こしたキャッシュ1808に対してスレッドを再発行を行う。二次キャッシュ1801あるいは外部メモリから取得した命令は、命令キャッシュ1808に格納されて、スレッドを再開する。次に同一の命令を実行する場合には、命令キャッシュ1808にすでに分岐先の命令が格納されていて、分岐のペナルティーも発生しない。

【0144】スレッド管理ユニット1807が、他の優先順位の高いスレッドが充満していて空きがない場合は、やはりスレッドの移住を行う。その場合は、スレッド管理ユニットからの通信で、スレッドの負荷の低いスレッド管理ユニット1809を探し出し、スレッドを移住させる。

【0145】この機構によって、同一命令を最大限に再利用することができる。さらに、従来のPMT方式と異なり、スレッドは命令アドレスにかかわりなく、自由に演算ユニットに分配できる。

【0146】本発明のプロセッサは、厳密な分岐命令にスレッドの移住が必要であるため、分岐命令の実行の頻発を避ける必要がある。分岐はマルチスレッドによって隠蔽は可能であるが、スレッドの発行能力には上限があるためである。

【0147】そのために、命令アドレスとは無関係に命令を配置する。格納される命令の順序は、確率的に命令が実行されると予測される順序である。そして、予測された分岐方向を示す分岐予測情報をキャッシュのタグメモリに配置する。分岐予測情報は演算ユニットに送られ、分岐命令の実行結果と照合されて不一致の場合はスレッドを停止させる。

【0148】キャッシュのタグメモリに次の命令アドレスを示す値を置くことで、分岐命令の実行前に隣接する命令キャッシュから命令を取得させることもできる。この機構によって、PC相対分岐だけではなく、レジスタの示すアドレスへの分岐を予測することもできる。

【0149】同時に、前述のデータフロー同期情報も命令キャッシュのタグメモリに格納する。これによって、同じ命令を利用する限りは、すべてのスレッドから1つの分岐予測、データフロー予測情報を共有することがで

きる。

【０１５０】図１３に、本発明の命令キャッシュにおけるタグメモリの構造を示す。命令キャッシュにはそれぞれ命令ごとに数ビットの分岐予測情報、あるいはデータフロー予測情報が格納されている。発行された命令が分岐命令の場合は、分岐予測情報として使用し、発行された命令がロードストア命令の場合は、データフロー予測情報として利用する。命令ごとの予測情報のビット幅は、実行ユニットの数から決定される。データフローユニットが目的とする実行ユニットを特定するためである。

【０１５１】また、分岐命令の実行とは独立して次の命令キャッシュのアドレスを特定するために、次の命令アドレスを示す値が格納されている。この値によって、条件分岐だけではなく、オブジェクト指向言語の仮想関数に代表される、レジスタ値への分岐も予測することができる。

【０１５２】（仮想記憶と同期）

【０１５３】仮想記憶ユニットは、可能であれば全ての演算ユニットから共有することが望ましい。理由は、複数のプロセスが共存する場合は、要求される仮想記憶のエントリの数も増大するためである。更に、仮想記憶ユニットが分散した場合は、その内容のほとんどが重複するためである。

【０１５４】本発明のプロセッサは、内蔵するキャッシュをすべて仮想空間で管理する。メモリへのアクセスの時だけ、物理アドレスに変換するためにグローバルＴＬＢを用いる。

【０１５５】仮想キャッシュは、複数のプロセスが共存するために、異なるプロセス空間のキャッシュをアクセスしない機構が必要になる。そのために、キャッシュのタグメモリにはプロセスＩＤの情報を持たせ、キャッシュヒットの確認ごとにプロセスＩＤの一致確認を行う。

【０１５６】（データフロー同期）

【０１５７】一般的にマルチスレッドの同期は、あるスレッドからの書き込みをトリガにして直接別のスレッドを起動する方式がもっとも単純かつ有効である。この方式はデータフロー方式とよばれ、プログラムモデルから見てもっとも単純な方式である。プログラム上では、あるアドレスへのデータライトを自動的に感知してスレッドを再開するように設定するだけである。

【０１５８】この機構の実装のために、仮想記憶とデータキャッシュに特別な機構を設ける。仮想記憶には、あるアドレスのライトアクセスがあった場合にスレッドを生起する情報を書き込んでおく。そのアドレスを含むデータメモリをデータキャッシュに読み込む際に、データフロー参照がある情報も同時に取得する。

【０１５９】データキャッシュ側には共有ビットを書き込むだけとなる。形としては、ＴＬＢのデータフロー同期エントリとデータを共有するという形になる。これによって、各キャッシュエントリにはデータフロー同期情報を持たせる必要はない。前述のディレクトリ共有機構で十分であり、ＴＬＢから二次キャッシュに向けてデータフロー同期の開始を伝達する。

【０１６０】厳密なメモリ共有機構では、ある時点でのメモリの状態は、どのプロセッサから見ても同じであることが要求される。ところが、この厳密なメモリ共有は、キャッシュの搭載や、メモリの階層分割によって現実には不可能になりつつある。そのため、近年ではプロセッサの仕様の方を変更し、同期命令前後のデータアクセスの順序だけを維持するように定義を変えた。プロセッサの種類によって細かい違いはあるが、基本的にはこれをルーズコンシステンシと呼ぶ。

【０１６１】本発明のプロセッサでは、同期命令は他の演算ユニットからのデータの書きこみを待ち、すべて到達した時点でスレッドを再開する。ところが、遠距離の演算ユニットには制御信号が即座に届かないため、同期命令までに実行されたストアかどうかの判定は厳密には不可能である。

【０１６２】そのために、同期命令における「同時」の定義を変更する。たとえ実時間では後に実行されたストア命令も、同期命令の再開までに伝達が間に合った場合には時勢的に前だとみなす。

【０１６３】そして、同期とは、ＰＭＴパイプラインを一周回分待ち合わせて、他のスレッドの、「同時」の時間以前に実行された全てのストアを受理するまで待つこととする。これによって、単体のスレッドの場合と同じく、全てのスレッドのデータ転送はスレッドの移動に間に合えば良い。パイプラインが一周した時点でスレッドを再開させるが、その時点では同期命令「以前」の全てのストアは実行され、再開地点以降のデータキャッシュに格納されている。

【０１６４】この方法によって、全てのスレッドにわたって、同期変数の前後のメモリアクセスの順序を保持することができる。なおかつ、同期中に他のスレッドの動作が可能になり、性能へのペナルティーも隠蔽できる。

【０１６５】図２５に本発明のプロセッサにおける同期の動作を示す。スレッドＢからのＳｔｏｒｅＡは、スレッドＡのＬｏａｄＡで読み込むことが出来る。スレッドＡのＳＹＮＣ命令より実時間的には後に実行されているにもかかわらず、ＳＹＮＣ命令の再開までにキャッシュの伝達が終了しているためである。仮想時間的に前かどうかの判断基準は、前のＳＹＮＣ命令のパイプラインの到達よりも早いかどうかで決定すれば十分である。こうして、複数のＳＹＮＣ命令間で、ＳＹＮＣ命令前後のデータ格納順序を保つことができる。

【０１６６】さらに、従来のプロセッサと異なり、ＳＹＮＣ命令で他のスレッドを止める必要はなく、ＳＹＮＣ命令の伝達もスレッドと同じ速度で伝達すれば十分である。

【0167】図26に、ソフトウェアモデルから見た同期の動作について示す。スレッドAのSYNC命令の前に実行されたスレッドBのStoreAは、仮想時間では前に実行されたSYNC命令のさらに前に行われているため、スレッドAから読み込むことができる。

【0168】スレッド間の同期は、同期変数へのアクセスに対して、明示的にOSのソフトウェアによるスケジューラを起動して管理することが多い。しかし、前述のデータフロー同期機構を自動的に利用すれば最も高速である。

【0169】具体的には、あるデータをロードする同期命令の実行によって、データフロー同期ユニットにそのスレッドの状態とロードアドレスを転送する。スレッドはその時点でスリープする。データのストアはデータフロー同期ユニットとディレクトリ方式キャッシュコヒーレンシによって判定されて、待ち状態のスレッドを直接起こすことができる。

【0170】（パケット制御信号）

【0171】既存のスーパースカラ、VLIW方式に代表される命令レベル方式では、信号は可能な限り速く伝達することを要求される。ところが、回路規模が大きくなるとそれは現実的には不可能になる。理由は主に3つある。まず、微細化が進むと、配線遅延の比率が大きくなる。さらに回路規模が大きくなると、回路間の配線が爆発的に増大する。さらに、周波数が高くなると、隣接する配線間のクロストークやグラウンドバウンスが問題となる。前者の対処としては、配線を短縮するか、配線間の距離を大きくとりシールドする必要が出てくる。後者の対処には、電源配線を配線に対して最適化して、電流ループの大きさを最小限にする必要がある。

【0172】それに対して、PMT方式は、隣接するユニットを除き、制御信号の伝達は多少の遅れが許される。ということは、長距離の信号伝達に使用される信号線を、複数の信号が共有することができる。こうして、長距離の配線の本数を最小限にする。

【0173】更に、長距離の配線は一気に送ってしまうのではなく、中継する回路で受け止めてシフトレジスタ的に順に送ることができる。こうして、1クロックの間で伝送するのはルーター間の距離だけで済み、制御信号が動作周波数の向上を阻害することは無くなる。中継のためのルーターやラッチの規模が大きくなるという欠点はあるが、それは半導体のプロセスの向上の恩恵をそのまま受けることが出来て、相対的な影響は少なくなる。

【0174】個々の配線を最小限の長さにして、信号伝送の多少の遅れを許容することにより、その配線のドライブを行うトランジスタの駆動電流を不必要に上げる必要がなくなり、信号の高周波成分の増加を抑制することができる。これはクロストークやグランドバウンスなどの抑制につながり、これらの対策に必要な回路の増加を防ぐこともできる。

【0175】遅延時間に関しては、PMT方式の隣接ユニット以外の転送レイテンシを許容する特性によって問題にならなくなる。こうして、並列度を維持し、回路規模を最小限に維持しながら周波数性能の向上を可能にする。

【0176】パケット制御信号は、データ転送などの目的ではアドレス、データとともに送られる。すなわち、アドレス、データを転送するパケットは、アドレスバス、データバスの空きをスレッドバッファで待ち合わせることになる。これによって、各バスのアービトレーションはパケットルーターが一括して処理することができる。

【0177】本発明のプロセッサは、命令キャッシュ、演算ユニット、外部インターフェースなどのユニットごとにパケットルーターを随所に配置し、遠距離の制御信号の伝達の中継を行う。パケットルーターには、複数のパケットルーターと送受信を行うためのバスを持ち、必要に応じてデータバスなどの補助的なバスを並行して設ける。

【0178】個々のパケットルーターは一意の番号を割り振られる。番号はスレッドの進行方向にあわせて昇順に割り振られ、付随するバス信号、伝達先のユニットによって一意にルーティングの方法も決定される。

【0179】このパケット制御信号によって、隣接するユニットを除く全てのユニットへの制御が行われる。

【0180】本発明のプロセッサにおけるパケットは、到達予定時間の情報をパケット情報に含む。この時間とパケットルーターの持つタイミングカウンタを照合することにより、パケットが予定通り伝達されているかどうかを判定する。

【0181】パケットが遅滞している場合は、並行して走るスレッドに対して即座にパイプラインストールを要求して、スレッドを止める。パケット遅延の例外処理を発行して、OSレベルのソフトウェアが対処を行ってスレッドを再開させる。

【0182】本発明のプロセッサは、各ユニットの内部状態を全ての演算ユニットから監視、改変することを可能にする。そのために、演算ユニットからの要求をパケットに変換し、パケットルーターを利用して順次伝達する。伝達先のユニットは、内部状態を含んだパケットを送信元の演算ユニットに伝送する。なお、ロードのためのレイテンシは無論マルチスレッドで隠蔽される。

【0183】（プロセッサ間通信）

【0184】本発明のプロセッサを複数利用する際に、本発明の内部の演算ユニットと同じように、プロセッサをリング状に連結すれば、プロセッサ間転送スループットを最大にすることができる。これによって、1つのスレッドは複数のプロセッサにわたって展開することができ、命令、データ共有の利点を最大限に生かすことができる。

【0185】だが、本発明のプロセッサの内部と同じく、データの転送にはパイプラインの隣接転送だけではなく遠距離の転送も考えられる。リング方式転送の欠点は遠距離に伝送するのが難しいという点である。そのために、遠距離の演算ユニット間同士でショートカットパスで伝送することは、全体の転送速度を大きく向上させる。

【0186】このような転送はレイテンシ時間が増大するものであるが、複数のプロセッサ間での通信はそれらの間のパイプライン全てを通過する時間で行われれば良いため、数十クロック以上のレイテンシが許される。このため、プロセッサ外の低速インターフェースには最適である。

【0187】本発明のプロセッサでは制御信号をパケット化しているため、同じ制御信号を複数のプロセッサに分配できる。ユニットを指定するための識別コードを拡張し、全てのプロセッサを一意に表現することで、マルチプロセッサに向けて自由に制御信号パケットを伝送できる。

【0188】本発明のプロセッサを複数利用する際には、個々のプロセッサに個別にメモリを接続する。各プロセッサがデータの実体の場所を特定するために、個々のプロセッサが持つ仮想記憶を利用する。この場合、仮想記憶のエントリはそれぞれコピーを持つことになり、キャッシュと同じ共有管理を行うことになる。そのために、仮想記憶には共有状態を示すビットを設ける。ただし、オリジナルは常にメモリに接続された仮想記憶となる。

【0189】仮想記憶の改変の際には、キャッシュのフラッシュと同時に、他の仮想記憶に改変を直接伝達する。改変を伝達された仮想記憶は、共有状態に応じてそれぞれキャッシュのフラッシュを実行する。

【0190】本発明のプロセッサ同士で、データの共有がある場合は、仮想記憶のページ単位でデータの共有情報を設定する。キャッシュラインごとのビットを持つことができないため、ページ全体が共有状態の場合はその都度内部キャッシュのタグにアクセスして確認する必要がある。

【0191】まず、プロセッサから外部にロードストア要求を行うケースについて述べる。まず、ロード命令では、キャッシュにエントリがない場合、あるいはTLBに対して共有状態が指定されている場合は、TLBを介してプロセッサ外部からデータを取得する。TLBにアクセスを行い、ローカルメモリではなく外部のメモリとデータを共有している場合は、プロセッサ外部にリード要求を出す。

【0192】ストア命令では、二次キャッシュにTLBへの共有状態が設定されていることにより、TLBへのアクセスを行う。共有状態に設定されている場合は、データのコピーの無効化（インバリッド）を通達する。

【0193】次に、プロセッサ外部からロード要求を受理した場合について述べる。受理した仮想アドレスに対して内部のTLBへのアクセスを行う。内部キャッシュで共有状態に設定されている場合は、内部のキャッシュに仮想アドレスでアクセスして、プロセッサ外部にデータを伝達する。

【0194】次に、プロセッサ外部からインバリッド要求を受理した場合も、同様に受理した仮想アドレスに対して内部のTLBへのアクセスを行う。内部キャッシュで共有状態に設定されている場合は、内部のキャッシュに仮想アドレスでアクセスして、内部キャッシュにインバリッドを伝達する。

【0195】なお、TLBのエントリがない場合は、OSによる仮想記憶処理を行う。

【実施例】

【0196】（第一実施例）

【0197】図1に、本発明の第一実施例を示す。101は本発明のプロセッサである。

【0198】命令発行ユニット102は、スレッド発行ユニット103、命令キャッシュ104を内蔵する。スレッド発行ユニット103は、命令キャッシュ104にプログラムポインタ値を伝達して、実行ユニット105に実行すべき命令を伝送する。

【0199】実行ユニット105は、4つの共有レジスタファイル106と、4つの16ビット演算ユニット107と、複数の特殊演算ユニット108から構成される、共有レジスタファイル106と16ビット演算ユニット107、および特殊演算ユニット108は、オペランドクロスバスで相互に接続されている。スレッドのレジスタ値などの状態は全て、隣接する実行ユニット105に伝送される。ただし、従来のPMT方式と異なり、1つのスレッドの状態は4クロックで転送される。末端に到達した状態は、スレッド状態信号132によって最初の実行ユニットに伝送される。実行ユニットからのスレッド生成、分岐発行は、分岐発行制御信号109、134によってスレッド発行ユニット103に伝達される。

【0200】一次データキャッシュ111は8つ搭載され、そのうちの4つが1つの実行ユニット105に接続されている。接続にはクロスバスが使用され、同時に4つの一次データキャッシュへの任意のアクセスを可能にしている。ただし、同じデータキャッシュへの複数のアクセスがかち合った場合には、1つのアクセスだけを行い、他のアクセスを行ったスレッドをストールさせる。なお、従来のPMT方式と異なり、4つの一次キャッシュはアドレス値によって特定でき、1つのスレッドからすべてのバンクに自由にアクセスできる。

【0201】4つの一次データキャッシュ111～114は、1つのアクセスバッファ115に接続され、隣接するライトバッファと、やはり隣接する二次キャッシュ

１１６へのデータのやり取りを行う。

【０２０２】二次キャッシュ１１６は、２つの一次キャッシュからのアクセスバッファ１１５、１３１と、ＴＬＢなどのに接続されたアクセスバッファ１１７から要求を受理する。二次キャッシュユニット１１６は一次キャッシュと異なり、命令もデータも格納する。そして、二次キャッシュも複数の要求を受理するために複数のバンクに分けられてる。

【０２０３】アクセスバッファ１１７は、二次キャッシュ１１６からの要求によって外部とのアクセスを行う際に、データのバッファリングを行う。

【０２０４】新規スレッド発行ユニット１２７は、割り込み信号１２６の入力に応じて、内部に蓄積した待機状態のスレッドを発行する。あるいは、実行ユニット１０５からの直接のスレッド生成要求によってスレッドを発行する。そのために、スレッド発行ユニット１２７は、スレッド発行ユニット１０３に向けてスレッド発行制御信号１３３を出力する。

【０２０５】グローバルＴＬＢ１２０は、仮想アドレス信号の物理アドレスに変換し、物理アドレスをローカルメモリインターフェース１２２に伝送する。外部バスは基本的には仮想アドレスであることに注意。

【０２０６】ローカルメモリインターフェース１２２は、グローバルＴＬＢ１２０からの要求に応じて、ローカルメモリバス信号１２３を通じて外部メモリとのデータアクセスを行う。Ｉ／Ｏもローカルメモリインターフェース１２２によってアクセスできる。

【０２０７】共有バスインターフェース１２４は、共有バス信号１２５を通じて他のプロセッサに対してデータを送受信する。共有バス信号１２５から受理されたローカルメモリアクセス要求に対して、グローバルＴＬＢ１２０でプロセッサ内部でデータを共有しているかどうかの判定を行う。

【０２０８】（第二実施例）

【０２０９】図５に、本発明の第二の実施例の模式図を示す。

【０２１０】５０１は本発明の第二の実施例のプロセッサである。命令発行ユニット１０２と、実行ユニット１０５と、４つの一次キャッシュ１１１、二次キャッシュ１１６は隣接して配置される。この組が全体に８つ配置されることで、この第二実施例のプロセッサは３２のスレッドを同時に動作させることができる。本発明のプロセッサには、ユニットの搭載数に上限はない。

【０２１１】この第二実施例の個々のユニットは、本発明の第一の実施例に搭載されているユニットとほとんど共通であり、ユニットの組み合わせがだけが異なる。

【０２１２】前段プロセッサ接続インターフェース５０２は、別のプロセッサからのデータ転送を受理する。実アドレスで要求されたアクセスを、ＴＬＢ１２０を用いて内部のキャッシュ、ローカルメモリで共有されているかどうかを判定する。

【０２１３】ＩＰユニット５０４はソフトウェアよりもハードウェアの方が効率が良い処理を行うためのユニットである。これらはそれぞれ演算ユニットの近傍に配置される。演算ユニットはＩＰユニットの出力データをソフトウェアで即座に整形するため、ＩＰとプロセッサ間の転送が最小限になる。

【０２１４】２つのローカルメモリインターフェース１２２は、二次キャッシュからのメモリアクセス要求を受理して同時にメモリとのアクセスを行う。アクセスの前にはグローバルＴＬＢ１２０を利用して物理アドレスへの変換を行う。

【０２１５】Ｉ／Ｏバスインターフェース５１０はプロセッサに直接接続されたローカルなＩ／Ｏへのインターフェースである。

【０２１６】新規スレッド発行ユニット１２７は、スレッド発行命令の要求に応じてスレッド発行を行うとともに、割り込み信号１２６、ソフトウェア例外などの要求に応じて休眠状態のスレッドを生起する。

【０２１７】この実施例のプロセッサは、通常のマルチスレッドプログラムを利用して、浮動小数点命令を含めて３２並列動作を可能にしながら、規模的には単一プロセッサの８倍強の素子数で実現できる。個々のキャッシュは小容量だが、全てのキャッシュの内容を全てのスレッドから共有することができるので、個々のスレッドが１つの高速大容量キャッシュを持つのに等しい。

【０２１８】（命令発行ユニット）

【０２１９】図６は、命令発行ユニット１０２の内部構造の模式図である。

【０２２０】パケットルーター６０１は、スレッドを制御する制御パケット信号６０３を受理し、このスレッド発行ユニットで受理可能であるかを判定する。

【０２２１】制御パケット信号の内容がスレッドの受理の場合は、プライオリティー選択ユニット６０４に制御信号を伝達する。制御信号の内容がキャッシュの直接制御の場合は命令キャッシュ制御ユニット６０５に制御信号を伝達する。制御信号の内容がローカルＴＬＢ制御の場合は、命令ローカルＴＬＢユニット６０７に制御信号を伝達する。さらに、スレッド移住の要求である場合は、スレッド移住ユニット６２０に制御信号を伝達する。

【０２２２】待ち状態スレッドが一杯などの理由でパケット制御を受理できない場合は、別の隣接するパケットルーターに、制御パケット信号６１９を通じて伝送する。

【０２２３】プライオリティー選択ユニット６０４は、待ち状態スレッドバッファ６１８の中から、実行可能状態でかつ最もプライオリティー値の高いスレッドを１つ選択する。ただし、キャッシュミスなどで実行できない状態のスレッドは選択されない。このプライオリティー

選択ユニット６０４は、待ち状態のスレッドの数に対して爆発的に規模を増大させるため、待ち状態スレッドバッファ６１８の数を増やしすぎないことが求められる。そのために、パケットルータ６０２では、待ち状態のスレッドを１つのスレッド発行ユニットに集中させない制御が行われる。

【０２２４】本実施例では、命令キャッシュだけは物理空間キャッシュとする。異なるプロセス空間に属する命令を共有するためである。命令キャッシュはキャッシュ制御ユニット６０５、キャッシュタグメモリ６０６、命令ＴＬＢ６０７、キャッシュデータメモリ６０８で構成される。

【０２２５】キャッシュ制御ユニット６０５は、スレッドごとの命令キャッシュアクセスを実行し、パケットルーター６０２を介して要求される命令キャッシュへの直接アクセスを実行する。さらに、パケットルーター６０２からのグローバルＴＬＢの改変によるＴＬＢ６０７のエントリの無効化も行うことができる。

【０２２６】キャッシュタグメモリ６０６には、全ての物理アドレスが格納され、さらに、分岐予測、データフロー予測情報が格納される。

【０２２７】スレッド状態制御ユニット６０９は、キャッシュのヒットチェックを行う。命令ＴＬＢ６０７によって変換された物理アドレスと、命令キャッシュタグ６０６の結果が一致すれば、キャッシュはヒットしたことになる。その場合は、４つ分の命令を命令メモリ６１６から取得して命令順序アライナ６１４に伝達して実行可能な状態にしておく。

【０２２８】スレッド状態制御ユニット６０９は、前の命令発行ユニットの出したスレッド状態信号６０８を受理する。前のスレッドよりも待ち状態のプライオリティーが高い場合は、無条件でスレッドを発行する。前のスレッドがない場合は、前に実行した命令と同じ命令を使うスレッドが待機状態であれば、待機状態のスレッドを優先して発行する。命令アドレスが一致しない場合は、キャッシュから取得しておいた命令を発行する。発行したスレッドの状態は、隣接する命令発行ユニットにスレッド状態信号６１５で伝達される。

【０２２９】命令順序アライナ６１４は、蓄積された４つのスレッドのそれぞれ４つの命令を、１クロックごとずらして出力する。命令の種類によって配置を変えるようなことはしない。

【０２３０】スレッド状態制御ユニット６０９は、内部に格納された現在のスレッドのＰＣと、新規に発行されるスレッドのＰＣを比較し、一致するようなば、命令順序アライナ６１４に蓄積された命令の再利用を要求する。こうして、スレッドは同一の優先順位である限り、同じ命令を使用するものが優先的に実行される。

【０２３１】スレッド移住制御ユニット６２０は、演算ユニットで発生した分岐、スレッド発行を示す分岐要求信号６１３を受理する。自身の命令キャッシュに格納されていない場合は、パケットルーター６０２からキャッシュの要求を行う。他に命令をすでにキャッシュした命令発行ユニットがあれば、スレッドの移住を行うためにスレッド状態をパケットルーター６０２に伝送する。

【０２３２】命令バス信号６１７には、二次キャッシュ１１６からリプレースされる命令が送られる。取得した命令は、命令キャッシュのデータメモリ６１６に格納される。取得した命令は、スレッドが空き次第即座に発行される。

【０２３３】命令メモリは、命令アドレスと独立した命令を順に格納することができる。そのため、予測された分岐先を含めた命令の動作順に格納される。この機構を実現するために、命令キャッシュタグメモリ６０６はすべてのアドレスビットを含み、キャッシュヒット時にすべてのアドレスの比較を行う。

【０２３４】この機構を使用すると、同じ命令を使用し、同じ分岐方向を採択するスレッドは、命令キャッシュに常にヒットすることになり、命令リプレース時間だけでなく、分岐ペナルティー時間すら削減することができる。この機構は、同一の動作をする大量のスレッドで最大の効果を発揮する。

【０２３５】なお、この分岐予測が的中したかどうかを確認するために、命令ＴＡＧメモリには予想される分岐方向のビットを持たせる。レジスタ内容への分岐については命令キャッシュタグメモリ６０６から発行された次の命令アドレスを使用する。命令キャッシュのインデックスは、直前の命令キャッシュのインデックス値を常に使用する。インデックス値の算出は、スレッド発行時、命令キャッシュミスヒット時にのみ行われる。

【０２３６】同一の構造のスレッドでは、スレッド内部で同じ命令間でデータの受け渡しが行われる場合が多い。ただし、すべてのスレッドで同じアドレスを利用してデータを受け渡す場合もあれば、レジスタに対する相対アドレスを使用する場合もある。スタック、ヒープを用いる一般的な場合では、むしろ後者が多い。そのような場合では、データキャッシュ間の転送が必要になる。

【０２３７】そのために、命令間でデータの授受があるという予測ビットを設ける。データフロー予測ビットは、その命令が書き込んだデータアドレスを、自動的に別のラインに送ることを可能にする。

【０２３８】データフロー予測ビットには、バリッドビットとともに、送り先の演算ユニットを示す「ライン番号」を格納しておく。データのアドレスではないことに注意する。

【０２３９】データキャッシュミスで、データの実体を検索する際にやってきたパケットは、一次データキャッシュのヒットを検出することで、一次データキャッシュに隣接する命令キャッシュに向かってデータフロー予測ビットを書き込んでいく。そのために、データキャッシ

ュミスパケットには、データキャッシュミスの発生した演算ユニットの識別番号が伝達される。

【０２４０】なお、１つのライト命令に対して、複数のリード命令が同じデータを参照する場合は、リード命令同士の転送となる。そのために、データフロー予測ビットはロード命令にも必要になる。

【０２４１】分岐命令とロードストア命令は同時に利用されないため、データフロー予測ビットと分岐予測ビットは共用され、命令デコード結果によって使い分けられる。

【０２４２】スレッド間で共有するデータが多い場合は、ＰＭＴ方式が優れる。それに対して、自身のスレッド内部の転送量が大きく、スレッド間で共有するデータが少ない場合は、ＳＭＰ方式が優れる。これらの双方の長所を取り入れるために、ＳＭＰ実行モードを設ける。

【０２４３】ＳＭＰ実行モードは隣接する命令発行ユニット１０２の負荷が高く、データキャッシュのトランザクションの負荷が高い場合に、同じスレッド発行ユニットで連続して１つのスレッドを管理するモードである。本発明の実施例では、１つのスレッド発行ユニットで４つのスレッドを動作できる。

【０２４４】ＳＭＰ実行モードでは、スレッド状態を隣接する命令発行ユニット１０２に伝達せず、次のＰＣアドレスを自身のキャッシュ制御ユニット６０５で利用する。キャッシュ制御ユニット６０５は、キャッシュから４命令を取得して、命令順序アライナ６１４に送る。

【０２４５】他の演算ユニット、キャッシュとのキャッシュコヒーレンシや同期は、ＰＭＴモードと同じ階層キャッシュコヒーレンシ機構を用いて行われる。すなわち、本発明のプロセッサは、ＳＭＰモードでは階層キャッシュの共有メモリマルチプロセッサそのものとして機能する。

【０２４６】（演算ユニット）

【０２４７】図７は、４並列実行ユニット１０５の内部構造を示す模式図である。

【０２４８】命令デコードユニット７０３は、命令発行ユニットから送られた命令７２７をデコードし、各演算ユニットを制御する。同時に、４つのプログラムカウンタをインクリメントする。分岐命令が実行された場合は、演算ユニットで算出された分岐後のプログラムカウンタを利用する。更新されたプログラムカウンタは、隣接する命令デコードユニットに伝達される。

【０２４９】双方の実施例では、実行ユニット内部には、レジスタファイル７０４を４つ搭載している。１つのレジスタファイルは４つの演算ユニットで共有される。そして同時に１つの演算ユニットに対してのみレジスタを供給する。

【０２５０】レジスタファイル７０４は、コンテキストスイッチに対応するために４つのバンクを持つ。そして、レジスタファイル７０４は、隣接レジスタ転送を４クロックで完了する。そのため、一般的なＲＩＳＣプロセッサと同じ３つのリードポートと、隣接転送用の２つのレジスタリードライトの機能を持つ。現在実行中のスレッドが３つのリードポートを利用している間、さらに２つのレジスタを出力し、隣接する４並列実行ユニット１０５に転送する。そして、隣接する４並列実行ユニット１０５内部の、レジスタファイル７０４のうちの利用されていないバンクが、２つのレジスタの値を受け取って書きこむ。

【０２５１】こうして、レジスタファイル７０４は、現行のスーパースカラプロセッサよりも少ないポート数で実現でき、アクセスのための遅延時間を増加させないで済む。

【０２５２】オペランドクロスバスイッチ７０６は、４つのレジスタファイル７０４の値を、それぞれの演算ユニットに分配する。３つのオペランドを持つレジスタを４組分配する。受理する演算ユニット側には４入力のセレクタが３つ配置される。

【０２５３】演算ユニットで算出された演算結果は、即座にレジスタファイル７０４に伝達することはない。演算結果フォワーディング７１７を利用して結果を利用する演算ユニットに伝達する。そして、レジスタファイル７０４への書き戻しは、オペランドショートカット信号７２２によって隣接する演算ユニット１０５のレジスタファイルに伝達される。

【０２５４】整数演算ユニット７０８は、フラグ判定、１６ビットの範囲内での算術、シフト演算、分岐アドレス生成などを行う。４並列実行ユニット１０５内部に４つ配置され、それぞれが独自にスレッドの命令実行を行う。

【０２５５】この整数演算ユニットは１６ビット程度の加算器、シフタ、そして１６ビットを超えた演算が行われたことを感知する回路で構成される。これは演算ラインごとに実装される。

【０２５６】１６ビットを超える桁の変更が発生する演算は、パイプラインをストールして、共有６４ビット演算ユニット７１０を利用して再計算を行う。６４ビット演算ユニットは１６ビットの４倍以上の回路が必要になるため、１６ビット演算ユニットとオーバーフロー検出回路の組み合わせを利用し、それを４つ搭載して代用しても全体の演算ユニット数、回路規模あたりの性能を増やすことができる。

【０２５７】この方法が全体の性能をかえって向上させることができるのは、本発明の方式がマルチスレッドによって十分な並列処理を行うことができるという前提による。ＶＬＩＷ方式などの命令レベル並列では、並列動作可能な命令が並列度より少ない場合が多く、このようなペナルティーは絶対に許されない。

【０２５８】分岐ユニット７２１は分岐予測の判定と、分岐の発行、およびスレッドの発行制御を行う。ただ

し、分岐アドレスを算出するのは整数演算ユニット７１８による。実際の分岐は４命令に１回程度の頻度で実行される傾向が強いので、４つのスレッドで共有される。

【０２５９】分岐ユニット７２１は、分岐予測情報との照合を行い、一致した場合は自身のアドレス情報だけを更新する。分岐予測非成立の場合は、スレッド発行ユニットに分岐要求を伝達すると同時に、別の待ち状態のスレッドに切り替える。コンテキストスイッチは即座に行われ、実行ユニットの待ち時間はない。

【０２６０】基本的に、分岐後の処理は直後の実行ユニットで実行される。分岐予測が的中する場合は、自動的に分岐後の命令が次の実行ユニットに伝送される。

【０２６１】ただし、キャッシュミスの場合は、キャッシュの共有状態を確認することで、すでに命令が格納されている実行ユニットを検索する。発見された場合は、その実行ユニットにスレッドを移住させる。基本的にスレッドの移住には全てのレジスタファイルの転送が必要となる。ただし、データのほうはデータキャッシュコヒーレンシ機構が自動的に働くので必要はない。

【０２６２】関数からのリターンの場合は、スレッドを呼び出し元の実行ユニットに移住させる。ただし、渡すレジスタは１つの返り値のみである。スタックの退避、復帰は自動的に行われるので転送は必要ない。

【０２６３】ＳＭＰモードは、直後の演算ユニットで待ちあわせているスレッドのプライオリティーが高く、さらに後続のスレッドの負荷が低い場合に発生する。空いた演算ユニットを有効に利用するための手段である。

【０２６４】レジスタ同期ユニット７２３は、レジスタ内容の隣接ユニットへの転送と、スレッドの移住のためのメモリへの自動読み書きを行う。

【０２６５】スレッドの移住は、１つのバンクのレジスタの内容をそっくり他のスレッドに入れ替える作業である。実施例１のプロセッサにおいて、スレッド移住には合計４クロックを要する。

【０２６６】スレッドの移住には、メモリを介してレジスタの値を伝達する。スタックポインタから利用すべきメモリアドレスを演算ユニット７０８で算出し、現在のレジスタをロードストアユニット７１３に送る。新規のスレッドに対しても、スタックポインタからアドレスを算出し、ロードストアユニット７１３から新規のレジスタセットを読み込む。レジスタ退避の際には、ロードストアユニット７１３のアドレスバスもデータ転送に利用する。同時に４つのスレッド移住を行うため、８つのレジスタを同時に転送する能力を有する。

【０２６７】浮動小数点加算ユニット７１９、浮動小数点乗算ユニット７１２は、整数演算ユニットと異なり、精度が常に一定であり、動作が細かく決定されているので、倍精度の演算ユニットの機能のすべてを実装する必要がある。ただし、浮動小数点命令の出現頻度を考慮して、１つの実行ユニット１０８ごとに、浮動小数点加算ユニット７１９と、乗算と加算を同時に行う浮動小数点乗算ユニット７１２が１つづつ配置される。

【０２６８】なお、これらのレイテンシの長い演算は、複数のスレッドが同時に利用する。演算中は、これらの共有演算ユニットの内部にスレッドの情報が格納され、結果の値とともに整数演算ユニットに伝達される。

【０２６９】除算ユニット７１８は、除算、平方根などの、時間のかかり、かつ出現頻度の低い浮動小数点演算を行う。除算、平方根の演算は乗算と異なり、現実的な規模でパイプライン化して高速化する手段はない。そのため、１つの演算あたり数クロックのスループット時間が必要になる。そして、１つの除算ユニット７１８は、除算命令の出現頻度を考慮して、４つのスレッドで共有される。

【０２７０】ロードストアユニット７１３は、４つのロードストア命令の実行を同時に行い、８ワード分の転送能力を持つ。４つの演算ユニット７０５からの要求を整理してロードストアを行うとともに、待ち状態のスレッドのロードを実行する機能を持つ。

【０２７１】バイト単位の転送をワードに符号に応じて拡張する操作、あるいはその逆もこのユニットで行われる。

【０２７２】ロードストアユニット７１３には、４つのデータキャッシュが接続され、アクセスを行うアドレスによって使い分ける。データのアクセスは、同時に複数のユニットのアクセスを可能にする。そのために、４つのアドレス、データバスを互いにクロスバ接続する。

【０２７３】同じ一次キャッシュへのアクセスがかちあった場合は、優先度の低いスレッドを停止して、ロードの実行を待つ。キャッシュミスの場合も同様である。

【０２７４】ロードの衝突、キャッシュミスによるスレッドの停止の場合には、停止したスレッドの代わりに、前に停止してロードの終了したスレッドを再開する。

【０２７５】なお、前にロード、あるいはストアしたデータと、同じアドレスを利用するロード命令が直後に存在する場合は、データキャッシュへのアクセスを行わずに、同じデータを渡す。通常のプロセッサのライトバッファと異なり、渡す対象は同一スレッドでなくても良い。この機構に９よって、同一の命令を利用するスレッドの連続動作させる際のデータキャッシュアクセスが最小限となる。

【０２７６】演算結果フォワーディングユニット７１７は、実行ユニット１０５内部の演算ユニット間のデータの受け渡しを行う。同時に、長時間演算ユニットを利用する必要のある命令では、隣接する実行ユニット１０５に途中経過のレジスタ値を渡す。この機構によって、除算などの時間のかかる命令をパイプライン動作させることができる。同一の除算などの命令を利用するスレッドが連続する場合のスループット性能を高めるためである。

【0277】(データキャッシュユニット)

【0278】本発明のプロセッサは、データキャッシュのスループット確保、遠距離の一次キャッシュ間の転送のために、階層キャッシュ構造を取る。さらに、スレッド間の仮想記憶機構の共有のために、データキャッシュは基本的に仮想アドレスとしている。

【0279】データキャッシュは大きなスループットを要求されるため、擬似的に複数の要求を受理する構造とする。そして、キャッシュ内部のデータの共有管理のために、ディレクトリ方式キャッシュコヒーレンシを採用する。ディレクトリ方式はデータアクセスのレイテンシに劣るが、複数のキャッシュの要求に対応しやすい方式である。ディレクトリ方式の詳細については、文献1のP679からの記載を参照のこと。

【0280】文献1:Computer Architechture a Quantitative Approach Second Edition
著者:John L Hennessy, David A Patterson
出版社:Morgan Kaufmann Publishers, Inc.

【0281】図8に、本発明の第一の実施例における一次データキャッシュ111、二次キャッシュ116の接続関係模式図を示す。

【0282】4並列実行ユニット105には、一次データキャッシュ111が4つ接続される。4つの一次データキャッシュ111は、すべて1つの二次キャッシュ116に接続される。なお、二次キャッシュはデータ、命令の双方を格納する。

【0283】803は一次データキャッシュのタグである。806は、二次キャッシュのタグである。

【0284】データキャッシュは仮想アドレス空間を利用し、複数のプロセスが混在するため、異なったプロセス空間のエントリが混在する。そのため、タグメモリ内部にはプロセス空間のIDが配置され、一致比較の時にアドレスとともに比較される。さらに、タグメモリ内部には共有先を特定する共有ビットを有する。

【0285】一次データキャッシュ111、二次キャッシュ116は、アドレスの下位で分割したバンクを持ち、隣接する転送は同時に、そして、連続するアドレスは複数のキャッシュバンクが同時にアクセスさせることを可能にする。二次キャッシュタグメモリ806、二次キャッシュデータメモリバンク807も、アドレスに対して分割され、複数のアクセスを同時に受理する。

【0286】データキャッシュ制御ユニット802は、キャッシュミスの場合に適切なキャッシュからデータを要求する。さらに、データ転送の要求に応じて、適切なキャッシュにデータを転送する。さらに、内部のキャッシュの共有状態を管理する。

【0287】実行ユニット105が一次データキャッシュ111への読み込みを行うケースについて説明する。一次キャッシュデータメモリ804からデータを読み込むと同時に、一次キャッシュタグメモリ803に対してアクセスを行う。一般的なキャッシュと同じく、タグメモリの読み出し内容が要求されたアドレスと一致しない場合、あるいはそのエントリが無効となっている場合、データキャッシュミスとする。その場合、スレッドに対してストールを要求し、二次キャッシュ116からデータを要求する。

【0288】実行ユニット105が一次データキャッシュ111への書きこみを行うケースについては、まず一次キャッシュタグメモリ803だけに対してアクセスを行う。アドレスが一致してかつ、該当するデータが共有状態に指定されている場合は、二次キャッシュ116に対してインバリッド要求を発行する。

【0289】一次データキャッシュタグ803には、2ビットの共有情報を含む。隣接する一次キャッシュへの共有状態と、それ以外のキャッシュとの共有状態である。

【0290】アクセスバッファ115は、一次データキャッシュ111から二次キャッシュ116へのアクセスが不可能である場合に、アクセス情報およびスレッドの情報を蓄積する。あるいは、二次キャッシュ116から一次キャッシュ111へのインバリッド伝達の蓄積にも用いられる。

【0291】アクセスバッファ115は、一次キャッシュ111からの隣接転送要求を受理し、二次キャッシュ116を通さずに隣接するアクセスバッファ131にデータを送信することも行う。

【0292】同時に、アクセスバッファ117からデータを受理して、二次キャッシュ116内部の共有状態を調べる。共有状態であれば、データを格納するか、該当する一次キャッシュ111に伝送する。

【0293】二次キャッシュ116は、一次キャッシュ111からのキャッシュアクセスを受理するとともに、隣接する二次キャッシュ、さらにメモリインターフェース、あるいは実施例には存在しないが三次キャッシュからの要求を受け、適切な相手にアクセス要求等を送出する。

【0294】なお、本発明のプロセッサでは、データの転送やインバリッドの伝達は、スレッドの伝送速度に間に合えば十分である。SMP方式と異なり、階層バス間の転送レイテンシは演算ユニットの稼動率とはほとんど関係がない。そして、インバリッド伝達の方向も常に一定であり、転送スループットの確保も可能になる。

【0295】図14に、本発明のキャッシュにおけるタグメモリの構造を示す。一次キャッシュ111、二次キャッシュ116はともに仮想空間であるため、タグアドレスの一致比較だけでは不十分であり、プロセスIDの一致の判定が必要である。

【0296】ディレクトリ方式キャッシュの実装のために、共有状態を示すビットを設ける。一次キャッシュは、隣接する一次キャッシュと、二次キャッシュの2つの転送先が考えられるため、2ビットの共有情報を利用する。

【0297】二次キャッシュタグ806には、6ビットの共有情報を含む。隣接する二次キャッシュへの共有状態と、4つの一次キャッシュへの共有状態4ビットと、三次キャッシュ、TLBユニットなどへの共有状態1ビットで構成される。

【0298】(仮想記憶機構)

【0299】仮想記憶機構は、内部表記のアドレス表記を実際のメモリアドレスに対応させ、内部表記のアドレス以上の実メモリ空間を扱うことを可能にする。また、複数のプロセス空間の間の保護、およびメモリに存在しないメモリ空間の判定を行う。この仮想記憶の変換を効率的に行うためのバッファが、TLBユニット120である。

【0300】本発明の方式では、この仮想記憶機構にも以下の特徴がある。

【0301】(1)TLBは演算ユニットのある一定の集団ごとにそれぞれ専属のものを置く。

【0302】(2)キャッシュは仮想アドレスとし、実際のメモリのアクセスの直前まで仮想空間の変換を行わない。

【0303】(3)TLBの改変は、キャッシュにコピーがあるにもかかわらず、TLBエントリのない状態を作り出す可能性がある。

【0304】(4)スレッド間の高速同期のための、データフロー同期の機構を提供する。

【0305】複数のTLBを所持する場合は、TLB間で互いにコピーを持たせることになる。だが、オリジナルのエントリは常にメモリバンクに専属の1つとする。そのため、TLBの改変の際は、常にメモリバンクに専属のTLBに対して行う。オーナーであるTLBは、共有しているすべてのTLBに向かってページの無効化(インバリッド)を伝達する。

【0306】図9に、本発明の実施例におけるTLBユニット120の内部構造の模式図を示す。

【0307】仮想アドレス902は、TLBタグメモリ903、TLBデータメモリ909に入力される。構造的にはセットアソシアティブのキャッシュと同じである。TLBタグメモリ903は仮想アドレス902の内容と比較器904で比較され、一致した場合のみTLBデータメモリ904の内容を使用する。本実施例では、4ウェイセットアソシアティブ方式で実装することでタグメモリ、データメモリを4つ使用して、TLBのヒット率を向上させている。まったくページが一致するものがない場合は、ページフォルト例外発生ユニット905がOSプロセスを起動する。

【0308】仮想アドレスに相当するページがTLBユニット901内部に存在する場合は、TLBデータメモリ904の内容のうちの1つが、物理アドレスとして選択される。変換されたアドレスは、物理アドレス信号906から出力される。

【0309】本発明におけるTLBにはもう一つの役割がある。それは、データフロー同期と呼ばれる、指定したアドレスへのデータアクセスを自動的に検出する機能である。TLBエントリメモリ909には、アドレスの完全な一致を比較するための仮想アドレスが格納されており、ページの一致によってデータフロー比較器908に伝達される。仮想アドレスが完全に一致した場合は、データフロー同期発生ユニット907によって、登録されたスレッドが生起される。一致比較のマスクビットによるアドレス領域の指定も可能である。

【0310】図15に、本発明のプロセッサにおけるTLBユニットのエントリを示す。通常のTLBと同じく、変換後の物理アドレス、ページごとの保護情報などを持ち、複数のプロセス空間を混在させるためのプロセスIDを持つ。

【0311】通常のTLBと異なるのは、二次キャッシュや他のプロセッサへの共有情報を6ビット格納していることと、データフロー同期のための一致比較アドレス、一致比較範囲のビットを持ち、さらに、データフローの検出で生起すべきスレッドIDを格納していることの2点である。

【0312】本発明においてTLBユニットは、キャッシュのディレクトリ共有情報を示す最上位のエントリでもある。二次キャッシュの全て、ローカルメモリ、そしてプロセッサ外部への共有を示すビットをそれぞれ持つ。

【0313】そのため、二次キャッシュ同士やメモリへのデータ転送や、二次キャッシュからプロセッサ外部へのインバリッド要求などは、まずはTLBに要求される。TLBでは、TLBエントリの持つ6ビットの共有情報に従って、4つの二次キャッシュ、プロセッサの持つローカルメモリ、及びプロセッサ外部に直接伝達される。

【0314】制御信号パケットルーター910は、TLBへの書きこみを受理するとともに、データフロー一致やページミスによる例外スレッドを発行し、スレッドパケット911に向けて伝達する。

【0315】(外部インターフェースユニット)

【0316】本発明のプロセッサは、複数のプロセッサを接続して利用するために以下の特徴を持つ。

【0317】(1)スレッドを自動的に複数のプロセッサに分配する。

【0318】(2)各プロセッサにそれぞれローカルメモリを接続する。

【0319】(3)各プロセッサ間のアクセスは仮想ア

ドレス空間とする

【0320】本発明のプロセッサでは複数のメモリを接続し、それらを全て1つのスレッドの仮想アドレス空間からアクセスすることを可能にする。

【0321】図10に、データキャッシュと外部を接続するTLB120、ローカルメモリインターフェース122、プロセッサ間インターフェースユニット124の接続関係の模式図を示す。

【0322】本発明のプロセッサにおいて、基本的には物理アドレスは、TLBユニット120とローカルメモリインターフェース122の間だけで利用される。物理アドレス専用信号109が相互に接続される。

【0323】本発明のプロセッサにおいては、割り込みは最優先プライオリティーを持つスレッドの発行として処理される。リアルタイム性能は、スレッド制御ユニットによるプライオリティー制御によって確保できる。本発明のプロセッサは、プライオリティーの高いスレッドにいつでも動作を移すことができるためである。

【0324】本発明のプロセッサはマルチスレッドを前提としているため、複数のプロセッサ間でスレッドを発行するのにソフトウェア上の追加はほとんど必要ない。少なくともユーザーレベルのソフトウェアでは無改造で複数のプロセッサにスレッドを分配できる。

【0325】マルチプロセッサインターフェース124は、メモリアクセスバス125とともに、制御パケットバス1007を有する。プロセッサ内部の制御パケット信号1012は、そのままプロセッサ外部に出力することができる。

【0326】マルチプロセッサインターフェース124は、TLBによって該当する仮想アドレスがプロセッサ間共有状態を示す場合に、内部からの仮想アドレスを共有バス信号125に対して出力し、スレッド状態1012を制御パケットバス1007に出力する。

【0327】本発明のプロセッサは、外部の共有バス信号125からの仮想アドレスの受信によっても、TLB120へのアクセスを行う。プロセッサ内部にデータのコピーがある場合は、TLB120のエントリが存在し、二次キャッシュへのアクセスによってデータの実体のあるキャッシュの場所も階層的に特定することができる。TLBのエントリが存在しない場合には、OSによる仮想記憶処理によって本来の物理アドレス、メモリバンクの所在を特定することになる。

【0328】（制御信号パケット）制御信号をパケット化して伝達する方式は、制御信号をエンコードすること、複数の経路の配線を共有することで、制御信号の配線の規模、長さを最小限にできる。さらに、複数の信号のタイミング制御を、同一の回路で行うことで単純にすることができる。その欠点は、伝達のためのレイテンシが劣ること、パケットを中継するパケットルータの回路規模が大きいことである。

【0329】ところが、本発明の方式では、即座に制御信号を伝達する必要があるのは隣接するユニットにかぎられる。それ以外の制御信号は、スレッドのパイプライン進行にあわせて伝達すれば十分である。すなわち、パケット制御方式の欠点であるレイテンシは問題ではなくなる。そして、パケットルーターの回路的な規模の増大も局所的なものであるため、配線短縮の効果の方が大きい。

【0330】図11に、個々のパケットルータの内部構造の模式図を示す。パケットルーターは以下の3つの機能を持つ。

【0331】（1）パケットに応じてユニットの制御を行う

【0332】（2）パケットの目的地、情報量に応じて、複数のパケットルーターのうちの1つを選択してパケットを送り出す。

【0333】（3）パケットのタイミングをチェックして、スレッドの進行に対して遅れていればスレッドをストールする。

【0334】1101はパケットルーターである。受信した1102パケット信号を、コマンドデコーダー1103が解釈する。パケットをこのパケットルーター1101で即座に利用する場合は、制御信号デコーダー1104にパケット信号を入力する。制御信号デコーダ1104は、デコード結果と、パケットルーターのあるユニットの状態信号1105に応じて、個別のステートマシン1106を動作させ、ユニットの制御をローカル制御信号1107で行う。制御信号デコーダ1104、ローカルステートマシン1106の構造はユニットごとに異なる。

【0335】パケットを中継する場合は、まず、タイミングチェッカ1112でパケットが時間どおりに到達しているかどうかを判定する。時間に遅れている場合はスレッドストール要求信号1111でパケットを要求したスレッドを停止する。パケットバッファ1108に蓄積する。パケットが時間以内に到達している場合は、パケット出力ユニット1110で複数のパケットバスのうち1つを選択してパケットを出力する。

【0336】パケットの送信先は最終的な送信先に応じて静的に決定できる。トラフィックに応じた動的な経路制御などを行うわけではないため、一般的なネットワークで行うような最適な経路制御の必要はない。

【0337】図12に、本発明の第一実施例におけるパケットルーターの配置を示す。

【0338】パケットルーターは大まかなユニット、バスバッファごとに設置され、ユニットの制御を行う。パケット制御信号線は隣接したパケットルーターの間のみに配線される。

【0339】パケット制御信号は、スレッドのパイプラインの進行に従って伝送される。たとえば、演算ユニッ

トからＴＬＢユニットへの書き込みを要求した場合は、キャッシュユニットのルーターを通過して伝達される。転送の中継に時間がかかるため、転送は数クロックを要する。ただし、転送の間に別のスレッドの動作が可能である。

【０３４０】この機構によって、最小限の配線で並列数に見合うだけの数の制御信号を送ることができる。

【０３４１】図２７に、制御パケット信号の例を示す。すべての制御パケットは、３２ビット程度のＣＰ（ＣｏｎｔｒｏｌＰａｃｋｅｔ）信号を持つ。

【０３４２】ＣｏｎｔｒｏｌＰａｃｋｅｔには、パケットの機能を示すＰａｃｋｅｔＣｏｍｍａｎｄ、パケットのパラメータを示すＶａｌｕｅＦｉｅｌｄを持つ。ＲｅｑｕｅｓｔｏｒＵｎｉｔは要求元、ＴａｒｇｅｔＵｎｉｔは伝達先のユニットを示す。

【０３４３】ＲｅｍａｉｎｉｎｇＴｉｍｅはパケットが時間内に伝達されたかどうかの確認を行うための値である。この値をデクリメントすることで、パケットの進行が間に合っているかどうかの判定を行う。ＵｓｅｒＬｅｖｅｌは、制御パケットの特権レベルを示す。

【０３４４】スレッドの情報が必要なパケットは、やはり３２ビット程度のＴＩ信号が付加される。ＴＩにはプロセス、スレッドＩＤと、スレッドの優先順位、ユーザーレベルが格納されている。

【０３４５】この２つに加えて、アドレス、データ、ＰＣ（プログラムカウンタ）、ＳＰ（スタックポインタ）などの値が付加される。ＴＩとＰＣ，ＳＰによって、スレッドのすべての情報が管理される。ＣＰとアドレス、データが通常の内部バストランザクションに利用される。なお、制御パケット信号の仕様は、スレッド状態転送、データ転送などの目的によって変えることもでき、共用することもできる。

【発明の効果】

【０３４６】（回路規模）

【０３４７】基本的に、プロセッサに求められる性能は周波数性能と並列性能の積である。しかし、利用目的によっては、コストあたりの性能、および消費電力あたりの性能も求められる。本発明の方式は、回路の組みあわせによってそれらのいずれにも最適な構成にできることを示す。

【０３４８】今後のプロセッサの速度は、配線遅延にほぼ比例して決定される。半導体のプロセスの進化に従い、回路の局所的な遅延時間は縮小傾向がある。しかし、それには配線もトランジスタと同じオーダーで縮小するという前提条件が必要である。そのため、回路の大規模化によって配線が縮小されない場合は、微細化にもかかわらず周波数性能の向上は不可能になる。そのため、チップ全体の配線を行わないようにして、配線のオーダーを増加させないことが、周波数性能の向上を維持するために不可欠である。

【０３４９】配線の規模は、データ転送幅と転送相手の数で決まる。データ転送幅のＮ倍の増加に対して、配線の規模はＮ倍に比例して増大する。遅延の増大はわずかである。それに対して、転送相手のＮ倍の増加に対しては、配線の規模はＮの二乗に比例して増加する。そして、遅延もＮ倍で増加する。そのため、転送相手を増やすことより、転送幅を増やすことの方が遥かに容易である。

【０３５０】本発明のプロセッサは、スペクトル階層化によって転送相手の組みあわせを常に４つ程度に制限している。この規模は現行のインオーダースーパースカラ方式プロセッサと同程度である。これ以上の一対一接続の配線は行わないため、いくら並列度が増加しても、周波数性能を阻害する配線長の増加が発生することはない。

【０３５１】シフトレジスタ型転送は、自由な転送能力、比較的高いピークバンド幅、バスのアービトレーションの容易さという長所があるが、欠点としてはレイテンシ性能が最低となる。このレイテンシ性能の低下を隠蔽するためにマルチスレッドを利用する。

【０３５２】表１に、ＶＬＩＷ方式、ＳＭＰ方式、本発明のＰＭＴ方式ごとの回路規模、遅延時間の比較の表を示す。

【０３５３】ＶＬＩＷ方式は、並列規模の増大に対して周波数性能を著しく低下させる。マルチプロセッサ方式は、周波数性能は維持できるが、回路規模の増大が大きい。それに対して本発明のＰＭＴ方式は、メモリ、演算ユニットの共有によって、最小限の回路規模で並列性能を増加させることができる。

【０３５４】（ユニット稼働率から見た性能向上）

【０３５５】本発明の方式は、単体のスレッドのレイテンシ性能ではほかのパイプラインプロセッサに劣るが、複数のスレッドの動作全体で性能を稼ぐことができる。そのため、全体の性能はすべてのスレッドの和である大域的な処理性能で判断されなければならない。さらに本発明の方式は、演算ユニットなどの稼動率を最大にすることで、回路規模に対する全体の性能を最大にできる。それに対して、ほかの方式の多くは回路規模を増加するほど演算ユニットなどの稼動率が下がる傾向がある。以下、演算ユニットなどの稼動率が他の方式に対して高いことを示す。

【０３５６】表２は、本発明の方式と、ＳＭＰ方式との各状況に対する演算ユニットの停止期間の比較の表である。

【０３５７】本発明の方式は、自由な命令配置能力と、局所ＳＭＰモード機能によって、あらゆる演算ユニットをほぼ常に動作させることができる。従来のＰＭＴ方式が命令の配置に命令アドレスの制限があったのと対照的である。

【０３５８】本発明のプロセッサは、例外の発生頻度がスレッド発行ユニットの供給能力を超えない限り、ほと

んど全てのペナルティーを隠蔽することができる。すなわち、スレッドが十分供給されている限り、本発明の方式はＶＬＩＷ、マルチプロセッサ方式よりも演算ユニットの稼動率で勝る。しかも、それはスレッドが独立に並列動作できる限り、演算ユニットの数に比例して性能を向上できる。

【０３５９】マルチプロセッサ方式は、コンテキストスイッチにＯＳの介在が必要である。さらに、スレッドを別のプロセッサに移して再開させる「スレッドの移住」に、すべての状態をキャッシュコヒーレンシで転送する必要がある。この動作には、約１００クロック以上の間プロセッサのバスを占有するため、数千クロック以上のレイテンシを隠蔽するのでなければ意味がない。さらに、スレッドを再開するには、動作しているスレッドがＯＳを呼び出して、各スレッドに対して再開条件が整っているかどうかを確認しなければならない。

【０３６０】次に、マルチプロセッサ方式にコンテキストスイッチ機能をハードウェアで実装して自動的に行うことを考える。それでも、すべてのプロセッサに大量のスレッド、そしてそれら全てのスレッドの状態と、完全なスケジューリングハードウェアを同時に搭載する必要がある。さらに、スレッドの移住には大量のプロセッサ間転送が必要になり、オーバーヘッドは削減できない。

【０３６１】以上の結論として、マルチプロセッサ方式とレイテンシ隠蔽機構は両立しにくい。

【０３６２】本発明の方式は、コンテキストスイッチはハードウェアで実装される。本発明の方式は、すべてのスレッドの空きスロットが１つのスケジューリングハードウェアを通るので、どの空き状態のノードにも即座にスレッドを供給できる。

【０３６３】さらに、本発明の方式では、停止していたスレッドは、基本的には停止した時と同じノードで再開することができる。この場合は状態の転送が一切必要なく、そのノードが空いた時点で即座にスレッドを再開できる。このため、スレッドの移住をほとんど行わずに、最適なスレッドの負荷分散が可能になる。

【０３６４】さらに、同期変数の待ち合わせに関しても、同期変数のアクセスに上ってバスを止めることはない。更に、データフロー同期を利用すれば、スケジューリングの必要も同期変数の確認も必要ない。この機能によって本発明のプロセッサはマルチスレッドのオーバーヘッドをなくし、マルチスレッドをあらゆるレイテンシの隠蔽に使用することを可能にしている。

【０３６５】本発明の方式における唯一のＳＭＰ方式に対する短所は、パイプライン間のスレッドの移住である。しかし、スレッドの移住の頻度はパイプラインを長くすることによって減少させることができる。

【０３６６】最小限のキャッシュ容量でスレッドの稼動率を上げるためには、同じ命令やデータを利用するスレッドを集中して実行すれば良い。それは同じ工程の仕事を集中して行うほうが効率が良いことを意味する。本発明の方式は、命令、データキャッシュミスの管理によって、ある程度は自然にこの共有の形になる。

【０３６７】本発明の方式は、隣接する演算ユニットに全てのレジスタ状態を転送するかわり、共有するデータの転送量が減るＰＭＴ方式を基本とする。それに加えて、レジスタ状態を転送しない代わりに、共有するデータの転送量が最大となるＳＭＰ方式も可能にする。

【０３６８】ＰＭＴ方式では、命令の間のデータ転送スループットは、近傍の命令間ほど多く必要とされ、命令間が遠距離になるほど減少する傾向にある。それに対して、スレッド間のデータ転送のためのスループットは、スレッド間のデータの共有が多く、並列度が増大するほど拡大する。理由は、１つのデータを大量のプロセッサがほぼ独立して参照するためである。そのために、ＳＭＰ方式ではメモリバス稼働率に著しい偏りが生じる。

【０３６９】ＰＭＴ方式は、スレッド間のデータ転送のスループットを最小限にする方式である。よって、どんなに並列度を上げて、かつ同じデータを共有しても、局所的なデータバスのデータスループットの増加を抑制することができる。すなわち、データの共有とスケーラブルな並列性能向上を同時に実現することができる。

【０３７０】それに対して、ＳＭＰ方式は、スレッド間の転送には弱いが、単体の演算ユニットだけで実行ができるという長所がある。そのため、独立したスレッドの実行では、ＳＭＰ方式を利用するほうがメモリバスの稼働率が最小となる。

【０３７１】本発明の方式は、データキャッシュ間の転送量によって自動的にＰＭＴ方式、ＳＭＰ方式を使い分け、常にメモリバス稼働率を最小にすることができる。

【０３７２】一般的に、キャッシュの容量が増大すれば、それだけキャッシュミスの確率が減少して全体の性能を上げることができる。しかし、キャッシュの容量の増大はキャッシュアクセスの速度低下を招く。そのためには、キャッシュを分割するのが望ましいが、複数のキャッシュへの接続はやはり配線遅延による速度低下を招く。理想的なキャッシュ容量増大の方法は、キャッシュと演算ユニットを直結させて、それを組にして大量に配置することである。しかし、従来のマルチプロセッサ方式では、キャッシュを複数持たせても、複数のキャッシュのほとんどに同じ内容を格納する必要があり、キャッシュの容量増大の効果を見込むことはできない。

【０３７３】それに対して、本発明の方式では、ＰＭＴ方式を利用する限りは、複数のキャッシュへの同じデータの複製を抑制することができ、キャッシュの容量を増大させてヒット率を向上させることができる。さらに、スレッドの中で何度も利用するデータ、あるいはスレッドの中で発生したデータについても、データを利用するキャッシュに対してのみ直接データを送るため、データの複製が最小限で済む。

【0374】(命令、データ、演算ユニットの共有)
【0375】現在のプロセッサでは、命令メモリの内容はプログラムのロード時に決定され、まず改変されることはない。それを許すと、命令の読み込み、動作順序が保証されない現在のプロセッサでは動作が保証されないためである。
【0376】そのため、命令メモリはアドレスに対して必ず同じ値が読み出され、他のスレッドからの改変のおそれもまずない。ということは、同じ命令メモリを利用するスレッドはすべて1つの命令を利用できれば効率的である。PMT方式の作用によって、1つの命令は連続してパイプライン状に動作する大量のスレッドから参照できる。そのため、本発明の方式はオンチップマルチプロセッサなどに比較して命令のメモリサイズ、リプレースに要求されるメモリスループットが遥かに小さい。
【0377】本発明の方式では、パイプラインの動作を止める分岐命令、データキャッシュミスは、マルチスレッドによってある程度は隠蔽できる。しかし、スレッド発行ユニットの供給能力を超えるほど頻発する場合は、本発明の方式でもやはりパイプラインを停止することになり、性能を低下させる。そのため、命令に置かれた予測情報を用いて、そのペナルティーを極力減少するのが望ましい。
【0378】本発明の方式は、1つの命令を全てのスレッドが共有できる。そして、分岐予測情報、データフロー予測情報は、命令列の内容、すなわち命令アドレスに依存し、個別のスレッドの状態にほとんど依存しない。ということは、これらの予測情報は1つあれば全てのスレッドから共有できる。
【0379】マルチプロセッサ間データ転送は、プロセッサの数が増加するにしたがって局所的にも増大し、個々のプロセッサのバス転送性能を使用し、マルチプロセッサにおいてスケーラブルな性能向上を阻害する。
【0380】プログラムで利用するデータには、細かい数値の相違はあるものの、80%の部分のプログラムで20%の部分のデータを利用するという経験則がある。たとえば、キャッシュはこの経験則を利用するものである。ということは、1つのプログラムを分割したスレッドも、その多くは同じデータを利用することになるのは当然である。この性質を利用するために、まったく違うスレッドが同じデータを利用する方法を提供する。
【0381】データは同じ命令が同じようなデータを利用するケース、あるいはまったく違うデータを利用する2つのケースが考えられる。当然データの共有の効率は命令ほどではないが、大きなデータブロックに対しては共有できるケースが多い。そのために、データキャッシュを分散配置し、複数のスレッドから共有させる。
【0382】これによって、データキャッシュの共有と大容量化を同時に実現し、結果的に単体スレッドから見た一次キャッシュの容量を増加させることができる。無論、一次キャッシュ間の転送量は増加するが、それはスループットのみの増大であり、比較的実装しやすい。
【0383】本発明の方式では隣接する4つ程度の演算ユニットが1つのスレッド発行ユニットを共有する。これは、分岐、例外によるスレッドの切り替えの頻度が数命令に一回という前提によるものである。
【0384】待ち状態のスレッドは、この4つの同時実行されているスレッドのうちの、どのスレッドが停止しても即座に発行できる。
【0385】さらに、演算ユニットの列の長さの増加、分岐予測などによってスレッドの移住の頻度が減れば、スレッド発行ユニットの稼働率も相対的に減少する。
【0386】そして、本発明の方式では細かいスレッドの切り替えのためのスレッドの移住も必要ない。レジスタ、データキャッシュの内容は、常にスレッドが停止した場所に待機されており、スレッドの空きスロットを待つだけで即座に実行を開始できる。
【0387】SMP方式では、キャッシュレイテンシ隠蔽のためには、すべてのプロセッサがそれぞれ実行可能な待ちスレッドを待機させておく必要がある。あるいは、隣接するいくつかのプロセッサに対してスレッド発行を行わせることになる。このことは、大量のプロセッサに対して任意のスレッドを高速に発行することが難しいことを意味する。
【0388】本発明の方式では、各スレッド発行ユニットを、すべてのスレッドがパイプラインとして通過することによって実行させる。このため、すべてのプロセッサが待ち状態のスレッドを有することなく高速コンテキストスイッチを可能にする。データキャッシュや特殊演算ユニットの結果などを取得し、再開する準備が整ったスレッドは、常に空いたあらゆるスレッドスロットに対して発行される。
【0389】(IPユニットの共有)
【0390】IPユニット間のデータの転送能力を最大にするには、IPユニット間を信号で直結するのが最も簡単である。だが、それでは全体で1つの機能しか実現できない。
【0391】次に考えられる手段は、それぞれIPユニットの間にマイクロプロセッサをそれぞれ置くことである。しかしこれでは、プログラムがIPの結合ごとに分散されることになり、処理が一様にならないという欠点がある。
【0392】さらに次に考えられるのは、IPユニットとマルチプロセッサをクロスバスイッチで結合する方法である。これならば、共有バスよりは優れた転送能力が確保できる。しかしクロスバスイッチは回路規模が(M個のプロセッサ、N個のIPユニットユニットに対して)MとNの積のオーダーでで増加する方式であり、大規模並列には向かない。さらに、そのために切り替えのレイテンシ時間が遅く、自由でかつ動的な転送には向か

ない。

【0393】本発明の方式は、各ノードに対してIPユニットを接続して、IPユニット間の通信能力をノード間のデータ通信能力で確保する。IPユニット間のデータの整形は、PMTの各演算ユニットがそれぞれ独立して行い、IPユニットへの入力の負荷が低い場合はすぐに別の用途に転用できる。

【0394】IPユニットの転送能力が単体の演算ユニットの転送能力を超える程ど高い場合には、近傍の複数の演算ユニットを利用して転送し、その先の演算ユニットでデータを整形することができる。このような場合では特に、マルチプロセッサ+クロスバスイッチ方式より圧倒的にIPユニットからの転送性能を稼ぐことができる。

【0395】本発明の方式では、IPユニットはソフトウェアでは特殊命令、あるいはシステムアクセス命令として使用することができ、その配置に制限はない。実際のIPユニットの分散配置に対しては、スレッドの移住機構が自動的に対応することもできる。IPユニット間のデータ転送は、PMT方式が持つレジスタ隣接転送、キャッシュコヒーレンシ機構で行う。こうして、同じソフトウェアで自由なIPユニットの組み合わせに効率良く対応することができる。

【0396】(消費電力予測)

【0397】CMOS回路は、信号の変化のときに電力を消費する。信号が変化しなければ電力をほとんど消費しない。

【0398】ところで、本発明の方式は、同一のスレッドを連続して動作させるときは、その供給される命令、演算ユニットの状態は完全に同一である。さらに、利用するレジスタファイル、データバス、データキャッシュとのバス通信の内容もスレッド間の違いは少ない。ということは、同じスレッドをまとめて実行する時には、各スレッド間のわずかな動作の違いだけが消費電力になる。それに対して、通常のプロセッサでは、各命令ごとにすべての回路の状態が変わるため、すべての回路の半分近くの信号が変化し、消費電力となる。

【0399】結論としては、本発明の方式のプロセッサは、同一の命令、データを利用したスレッドの連続動作が可能な場合は、現行のパイプライン方式プロセッサよりも低い消費電力で同じ性能を発揮できる。アーキテクチャのレベルでこれ以上の低消費電力の手段は考えられない。

【0400】表1の記載のように、本発明のプロセッサは、マルチプロセッサ、VLIW方式に対して、性能に対する回路規模が最小である。理由は、PMT方式は命令、データ、演算ユニットの共有を行うためである。性能に対する回路規模が最小であるということは、そのまま性能に対する消費電力が最小であるということを意味する。

【0401】さらに、本発明の方式は、性能に対する配線長も最小である。今後の半導体の消費電力は、配線容量の充放電が大半を占めることになると予想されるため、配線が最小であるということはそのまま消費電力の削減に繋がる。

【0402】さらに、前述した同一命令を利用するスレッドの連続動作による電力削減といまって、本発明の方式は、プログラム可能な回路において、最小の電力で実際の演算を行う方法であるといえる。ただし、本発明の方式は局所的にはSMP方式に近い動作モードも持つため、その部分はSMP方式と同じ消費電力になる。しかし、本発明の方式は可能な限りPMT方式で演算を行おうとするため、演算性能に対する消費電力は常に最小になる。

【図面の簡単な説明】

【図1】本発明の構造を用いたプロセッサの構造模式図(第一実施例)

【図2】従来のVLIW方式のプロセッサの構造模式図

【図3】従来のマルチプロセッサ方式のプロセッサシステムの構造模式図

【図4】従来のPMT方式のプロセッサの構造模式図

【図5】本発明の構造を用いたプロセッサの構造模式図(第二実施例)

【図6】命令発行ユニットの内部構造模式図

【図7】最大4つのスレッドを同時に実行する、実行ユニットの内部構造模式図

【図8】一次、二次キャッシュの接続関係を示す構造模式図

【図9】TLBユニットの内部構造模式図

【図10】TLBと外部インターフェースの接続関係を示す構造模式図

【図11】パケットルーターの内部構造模式図

【図12】本発明の第一実施例における、パケットルーターの配置図。

【図13】命令キャッシュタグメモリの1ラインごとの内容

【図14】データキャッシュタグメモリの1ラインごとの内容

【図15】TLBユニットの1エントリごとの内容

【図16】従来のマルチプロセッサにおける、スレッドの動作例

【図17】本発明のプロセッサにおける、スレッドの動作例

【図18】分岐命令実行における、命令発行ユニットの選択方法を示す概念図

【図19】命令キャッシュのもつ予測情報の書き込み、利用方法を示す概念図

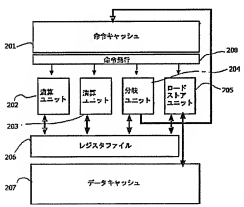
【図20】各種分岐命令の実行概念図

【図21】1つの演算ユニットにおける、パイプライン動作概念図

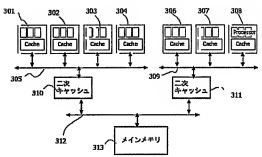
【図22】スレッド移住のレジスタ同期動作概念図
【図23】ディレクトリ方式キャッシュのリード動作概念図
【図24】ディレクトリ方式キャッシュのライト動作概念図
【図25】同期命令の動作概念図
【図26】同期命令のソフトウェア上での動作概念図
【図27】パケット制御信号の内容
【符号の説明】
101 本発明の第一実施例のプロセッサ
102 命令発行ユニット
103 スレッド発行ユニット
104 命令キャッシュメモリ
105 実行ユニット
106 共有レジスタファイル
107 16ビット演算ユニット
108 共有演算ユニット
109 分岐発行制御信号
110 データアクセスバス信号
111 一次データキャッシュ
112 アクセスバッファ
113 一次データキャッシュ
116 二次キャッシュメモリ
117 アクセスバッファ
120 グローバルTLB
121 データアクセスバス信号
122 ローカルメモリインターフェース
123 ローカルメモリバス信号
124 外部バスインターフェース
125 外部バス
126 割り込み信号
127 新規スレッド発行ユニット
131 アクセスバッファ
132 スレッド状態信号
133 スレッド発行制御信号
134 分岐発行制御信号
201 命令キャッシュ
202、203 演算ユニット
204 分岐ユニット
205 ロードストアユニット
206 レジスタファイル
207 データキャッシュ
208 命令発行ユニット
301、302、303、304、306、307、308 プロセッサ
305、309 一次共有バス
310、311 二次キャッシュ
312 共有メモリバス
313 メインメモリ
401 PMT方式プロセッサ
402 命令発行制御
403 PCラッチ
404 命令メモリ
405 演算ユニット
406 データバスクロスバスイッチ
407 データメモリ
408 外部インターフェース
409 演算要素
501 本発明の第二実施例のプロセッサ
502 前段外部プロセッサインターフェース
503 ショートカットバスインターフェース
504 IPユニット
510 I/Oバスインターフェース
511 次段外部プロセッサインターフェース
602 パケットルータ
603 制御パケット信号
604 プライオリティー選択ユニット
605 命令キャッシュ制御ユニット
606 命令キャッシュタグメモリ
607 命令ローカルTLB
608 スレッド状態信号
609 スレッド状態制御ユニット
610 スレッド状態信号
611 分岐、データフロー予測信号
612 命令信号
613 分岐要求信号
614 命令順序アライナ
615 スレッド状態信号
616 命令キャッシュデータメモリ
617 命令リプレースバス
618 待ち状態スレッド状態バッファ
619 制御パケット信号
620 スレッド移住制御ユニット
702 プログラムカウンタ信号
703 命令デコードユニット
704 レジスタファイル
705 レジスタ転送バス信号
706 オペランド転送クロスババス
707 オペランドショートカット信号
708 16ビット整数演算ユニット
709 結果ショートカットバス信号
710 64ビット整数演算ユニット
712 浮動小数点加算+乗算ユニット
713 ロードストアユニット
714 アドレスバス信号
715 データバス信号
716 レジスタ待避バス信号
717 演算結果フォワーディングタニット
718 浮動小数点除算ユニット
719 浮動小数点加算ユニット

720 結果ショートカットバス信号
721 分岐ユニット
722 オペランドショートカット信号
723 レジスタ同期ユニット
724 レジスタ転送バス信号
725 プログラムカウンタバス信号
726 分岐発行パケット信号
802 一次キャッシュ制御
803 一次キャッシュタグメモリ
804 一次キャッシュデータメモリ
805 二次キャッシュ制御
806 二次キャッシュタグメモリ
807 二次キャッシュデータメモリ
902 仮想アドレス信号
903 TLBタグメモリ
904 アドレス比較器
905 ページフォルト発生ユニット
906 物理アドレス信号
907 ページトラップ・データフロー同期発生ユニット
908 TLBエントリメモリ
909 制御信号パケットルータ
910 ページフラッシュシーケンサ
911 スレッドパケット
1001 データバス信号
1004 スレッドパケットバッファ
1007 スレッドパケット信号
1009 物理アドレス信号
1011 制御パケット信号
1012 スレッド発行パケット信号
1013 仮想アドレス
1101 制御パケットルータ
1102 制御パケット信号
1103 制御コマンドデコーダ
1104 制御信号デコーダ
1105 ローカル状態信号
1106 ローカル制御ユニット
1107 ローカル制御信号
1108 制御パケットバッファ
1109 制御パケット信号
1110 制御パケット出力ユニット
1111 スレッドストール信号
1112 制御パケットタイミングチェッカ
1201～1211 制御パケットルータ
1801 二次キャッシュ
1802、1804、1807、1809 スレッド管理ユニット
1803、1805、1808 命令キャッシュ
1806 分岐ユニット
1901、1905 命令キャッシュ
1902、1906 実行ユニット
1903 分岐ユニット
1904，1908 データキャッシュ
1907 ロードストアユニット

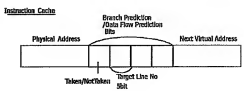
【図2】



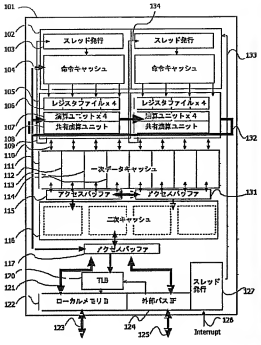
【図3】



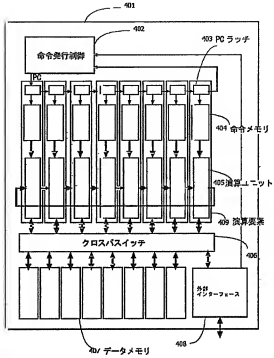
【図13】



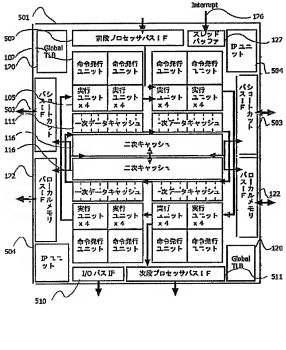
【図1】


スレッド発行
スレッド発行
命令キャッシュ
命令キャッシュ
レジスタファイル x 4
レジスタファイル x 4
演算ユニット x 4
共有演算ユニット
演算ユニット x 4
共有演算ユニット
一次データキャッシュ
アクセスバッファ
アクセスバッファ
二次キャッシュ
アクセスバッファ
TLB
スレッド発行
ローカルメモリ
外部バス
Interrupt

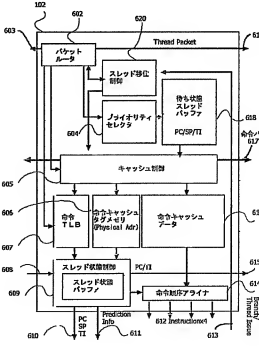
【図4】


命令発行制御
PC
PC ラッチ
命令メモリ
演算ユニット
演算要素
クロスバスイッチ
外部インターフェース
データメモリ

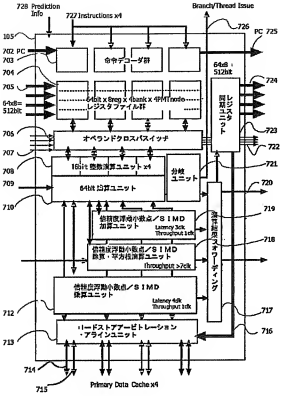
【図5】


Interrupt
前段ノードプロセッサバス I/F
スレッドバッファ
IP ユニット
Global TLB
命令発行ユニット
実行ユニット x 4
バリアレジスタ
一次データキャッシュ
二次キャッシュ
ローカルメモリ
IP ユニット
I/O バス I/F
次段プロセッサバス I/F
Global TLB

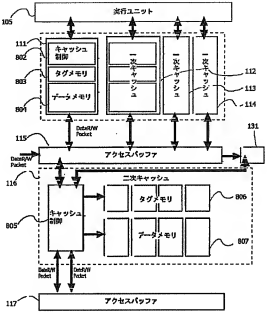
【図6】


Thread Packet
パケットルータ
スレッド移住制御
待ち状態スレッドバッファ
ノイオリティセレクタ
PC/SP/TI
命令バス
キャッシュ制御
命令TLB
命令キャッシュタグメモリ (Physical Adr)
命令キャッシュデータ
スレッド状態制御
PC/TI
スレッド状態バッファ
命令処理アライナ
PC SP TI
Prediction Info
Instructions
Security Thread Status

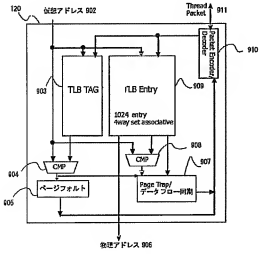
【図7】


728 Prediction Info
727 Instructions x4
Branch/Thread Issue
726
PC 725
103
702 PC
703
704
705
64bit= 512bit
706
707
708
709
710
命令デコード部
64bit x 8reg x 4bank x 4Port mode レジスタファイル群
64bit 512bit
レジスタ間転送バイパス
723
722
721
720
719
718
717
716
712
713
714
715
オペランドクロスバスイッチ
16bit 整数演算ユニット x4
64bit 整数演算ユニット
分岐ユニット
倍精度浮動小数点ノンSIMD加算ユニット
Latency 3clk Throughput 1clk
倍精度浮動小数点ノンSIMD乗算・平方根演算ユニット
Throughput 2clk
倍精度浮動小数点ノンSIMD除算ユニット
Latency 4clk Throughput 1clk
ロード&ストア・ユニット
ロードストアアービトレーション・アラインユニット
Primary Data Cache x4

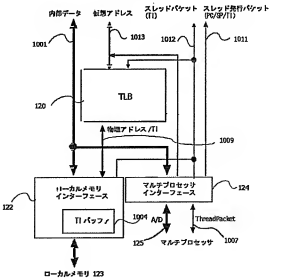
【図8】


105
実行ユニット
111
802
803
804
キャッシュ制御
タグメモリ
データメモリ
一次キャッシュ
一次キャッシュ
112
113
114
115
131
Data R/W Packet
アクセスバッファ
116
二次キャッシュ
805
キャッシュ制御
タグメモリ
データメモリ
806
807
117
アクセスバッファ

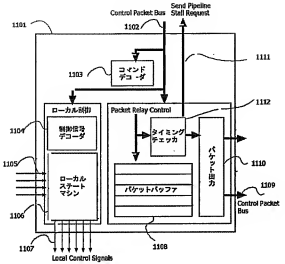
【図9】


120
仮想アドレス 902
Thread & Packet 911
Packet Encoder/Decoder
910
902
TLB TAG
TLB Entry
909
1024 entry 4way set associative
908
907
CMP
CMP
904
905
ページフォルト
Page Trap/データフロー制御
物理アドレス 906

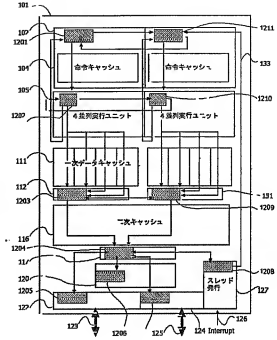
【図10】


内部データ
仮想アドレス
スレッドパケット (TI)
スレッド発行パケット (PO/IP/TI)
1001
1013
1012
1011
TLB
120
物理アドレス /TI
1009
ローカルメモリインターフェース
マルチプロセッサインターフェース
124
122
TI バッファ
1004
A/D
Thread/Packet
125
マルチプロセッサ
1007
ローカルメモリ 123

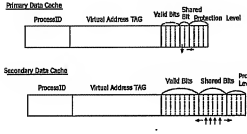
【図11】



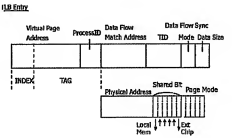
【図12】



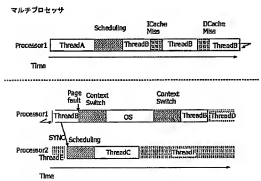
【図14】



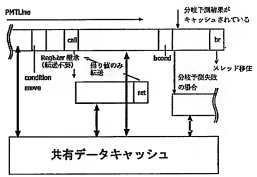
【図15】



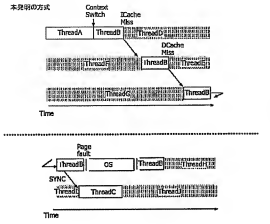
【図16】



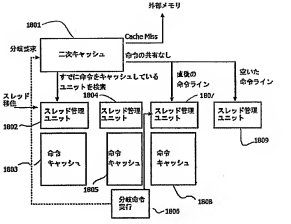
【図20】



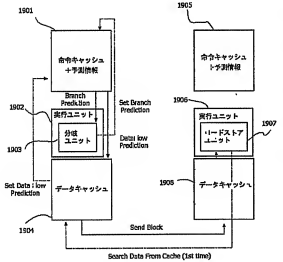
【図17】



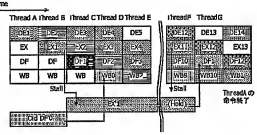
【図18】



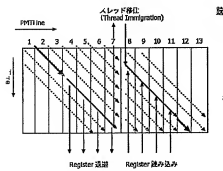
【図19】



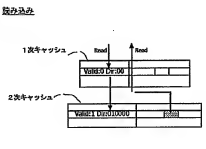
【図21】



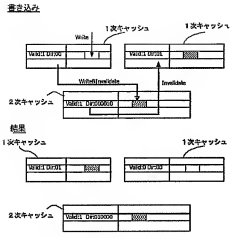
【図22】



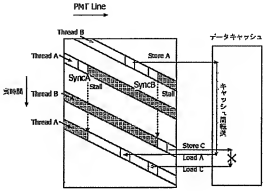
【図23】



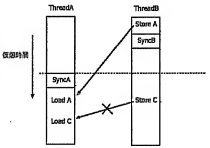
【図24】



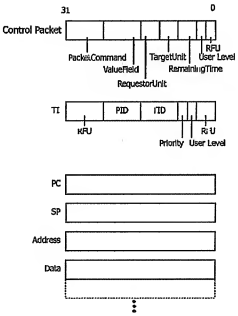
【図25】



【図26】



【図27】



フロントページの続き

(51)Int.Cl.⁷	識別記号	FI		(参考)
G06F 9/34	350	G06F 9/34	350B	
9/46	360	9/46	360B	
12/08		12/08	F	
			G	
			H	
			E	

	310		Y 310B
12/10 12/12		12/10 12/12	A A

Fターム(参考) 5B005 JJ13 KK13 LL01 LL11 MM02
MM03 MM32 NN31 PP21 UU32
5B013 AA01 AA05 AA11 BB01 BB18
CC06 CC13 DD04 DD05
5B033 AA02 AA03 AA04 AA13 AA14
AA15 BE05 CA01 CA09 DA04
DA14 DA17 DB02 DB03 DB06
DB12 DD01 DE07
5B098 AA02 AA10 DD01 DD03 FF01
GA05 GC03 GD02 GD03 GD12
GD14 HH07